NetSim Help - Index

NetSim – Versions and Components Overview …..Page (3)

NetSim – Getting Started …..Page (7)

NetSim File Organization …..Page (12)

Running Simulation via Graphical User Interface (GUI) …..Page (14)

- IP Addressing in NetSim …..Page (29)
- Applications …..Page (30)
- Performance Metrics …..Page (40)
- Dynamic Metrics (Available in Standard and Pro versions) …..Page (42)
- Generating Packet Trace (Available in Standard and Pro versions) …..Page (43)
- Generating Event Trace (Available in Standard and Pro versions) …..Page (47)
- Packet Animation …..Page (53)
- Saving Experiments …..Page (55)
- Opening and Deleting saved Experiments …..Page (57)
- Analytics …..Page (60)

Running Simulation via Command Line Interface (CLI) …..Page (62)

- Configuration File …..Page (67)
- Troubleshooting …..Page (71)

Writing custom code in NetSim (Available in Standard and Pro versions) …..Page (82)

- How to build dynamic link libraries (dlls)? …..Page (82)
- How to link dynamic link libraries (dlls)? …..Page (86)
- Hello World! Program …..Page (89)

How to debug Custom Code? …..Page (90)

Writing line number and file name of code in Event trace …..Page (94)

View/Accessing variables …..Page (96)

NetSim APIs …..Page (104)

Advanced Features …..Page (106)

Random number generator and seed values …..Page (106)

How to fail a Node in NetSim …..Page (107)

Static Routing …..Page (108)

NetSim - Batch Processing …..Page (111)

SINR, BER and Propagation models for 802.15.4 …..Page (117)

SINR, BER and Propagation models for 802.11 a,b,g and n …..Page (120)

WLAN 802.11n/ac …..Page (126)

Real Times Frame Capture (Available in Academic version only) …..Page (133)

Programming Menu (Available in Academic version only) …..Page (139)

Working Architecture …..Page (140)

How to create an executable …..Page (141)

How to De-bug your code linked to NetSim …..Page (151)

User Accounts …..Page (329)

NetSim Emulator (Available only with Emulator Add on Module) …..Page (331)

Note:

To enable Bookmarks in Adobe Reader, go to View →Show/Hide→ Navigation Panes → Bookmarks.

NetSim – Versions and Components Overview

NetSim is used by people from different areas such as academics, industry and defence to design, simulate, analyze and verify the performance of different networks.

NetSim Versions

NetSim comes in three versions, **Standard**, **Pro** and **Academic**. The standard version is used for project work and research while the academic version is used for lab experimentation and teaching. Pro version addresses the needs of defense and industry. The standard and pro versionsare available as 8 (eight) components in NetSim v8 from which users can choose and assemble. The academic version is available as a single product and includes all the technologies shown below. The main difference between standard and academic is the availability of protocol source code, and packet tracing facility in the standard version

Component No	Networks / Protocols	International Standards
Component 1	**Internetworks** Ethernet - Fast & Gigabit Address Resolution Protocol WLAN - 802.11 a, b, g , n, ac and e Propagation - Free space, Log-normal Shadowing, Rayleigh Fading IP v4 with VPN Firewalls Routing - RIP, OSPF Queuing - Round Robin, FIFO, Priority TCP, UDP **Common Modules** Applications (Traffic Generator): Voice, Video, FTP, Database, HTTP, Email, Peer-to-peer and Custom Virtual Network Stack Simulation Kernel Command Line Interface Metrics Engine with packet and event trace Packet Animator	 IEEE 802.3 RFC 826 802.11 a/b/g/n/ac/e RFC 2453,2328 RFC's 793, 2001 and 768

Component 2	**Legacy Networks** Aloha - Pure & Slotted CSMA/CD Token ring Token bus ATM X.25 Frame Relay Real Time (Frame Capture)	 IEEE 802.3 IEEE 802.4 IEEE 802.5 ATM Forum ITU Forum
Component 3	**BGP Networks & MPLS Networks** Border Gateway Protocol (BGP) Multi-Protocol Label Switching (MPLS)	 IETF RFC 1771 IETF RFC's 3031 and 3121
Component 4	**Advanced Wireless Networks** MANET - DSR, AODV, OLSR, ZRP Wi-Max	 IETF RFC 4728, 3561, 3626 IEEE 802.16d
Component 5	**Cellular Networks** GSM CDMA	 3GPP, ETSI, IMT-MC, IS-95 A/B, IxRTT, 1x-EV-Do, 3xRTT
Component 6	**Wireless Sensor Networks & Personal Area Networks** WSN with agent model & battery models ZigBee	 IEEE 802.15.4 MAC MANET in L3
Component 7	**Cognitive Radio Networks** WRAN	 IEEE 802.22
Component 8	**Long Term Evolution** LTE	 3GPP

Note: NetSim v7 had 7 components and component 8, LTE was added in NetSim v8. NetSim v6 had 16 components and the 16 component table of v6 is given below:

Component No	Protocols	Standards
Component 1	Aloha, Slotted Aloha, Token Ring, Token Bus, CSMA/CD	IEEE 802.3, 802.4, 802.5
Component 2	Net Patrol, LAN Speedometer	N/A
Component 3	Wireless LAN	IEEE 802.11 a/b/g
Component 4	Ethernet Switching, Gigabit Ethernet	IEEE 802.1 d
Component 5	X.25	ITU Forum
Component 6	Frame Relay	ITU Forum
Component 7	ATM	ATM Forum
Component 8	TCP, UDP	RFC's 793, 2001 and 768
Component 9	IP, Routing – RIP, OSPF, BGP	RFC 1058,2328, 1771
Component 10	MANET – DSR	IETF RFC 4278
Component 11	Wi-Max	IEEE 802.16 D
Component 12	Multi Protocol Label Switching (MPLS)	IETF RFCs 3031 and 3121
Component 13	GSM	3GPP, ETSI
Component 14	CDMA	IMT-MC, IS-95 A/B, 1xRTT, 1x-EV-Do, 3xRTT
Component 15	WSN	IEEE 802.15.4 MAC, IETF RFC 4278
Component 16	ZigBee	IEEE 802.15.4 MAC, IETF RFC 4278

What's new with NetSim v8?

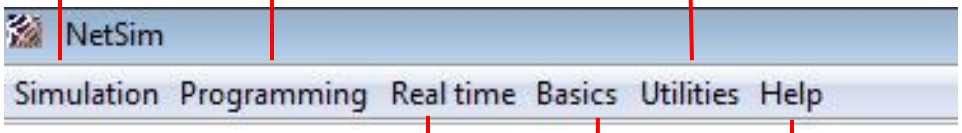
- Modeling and simulation of **LTE networks** as per 3GPP Standard
- **Dynamic Metrics** enables users to plot the value of a parameter over simulation time.
- **GSM and CDMA** have been integrated with NetSim's virtual stack. Users can now model the mobile switching centre (MSC) also.
- BGP integration with stack: BGP protocol has been integrated with NetSim's virtual network stack
- NetSim's C source code is now based on **Doxygen,** the de facto standard tool for generating **source code help**
- The wireless suite covered in NetSim is 802.11 a, b, g, n and **802.11 ac (Gigabit Wi-fi)**
- **P2P (bit-torrent), HTTP and E-Mail** application models have been added
- **File based mobility model** for MANET networks
- The IP layer of Inter Network component of NetSim now includes **VPN, Firewall and ICMP**
- Router to router links now support **data rates upto 100 Gbps**
- Network **Device properties** can now be user **customized**
- **Visual studio based solution files** for very easy custom code development
- **Multiple applications** can now be set from any source
- **Start time and end time** can be now set for any application

Main Menu

Opens the Simulation menu consisting of New, Open and Delete. User can Analyze, different Internetworks, Legacy, Cellular, BGP, MPLS, Advanced Wireless Networks, Wireless Sensor Networks, Cognitive Radio Networks and LTE Scenarios

Opens the Programming menu where different network programming exercises are available. Users can link and run their source code here. ***Note: This menu is available only in Academic and Standard Version.***

Menu to create users, set passwords, and sample / exam mode. Switching of users can be done through the login as option.



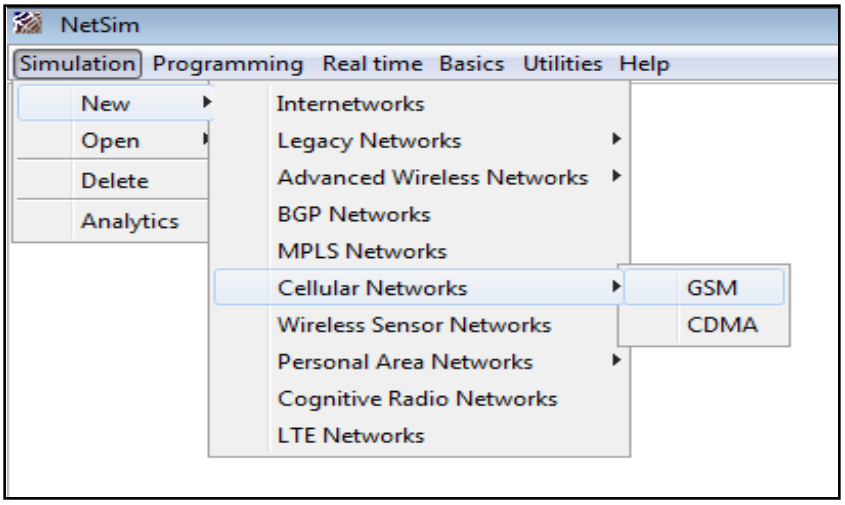
Displays all the Help related to NetSim. Help covers Simulation Experiments also.

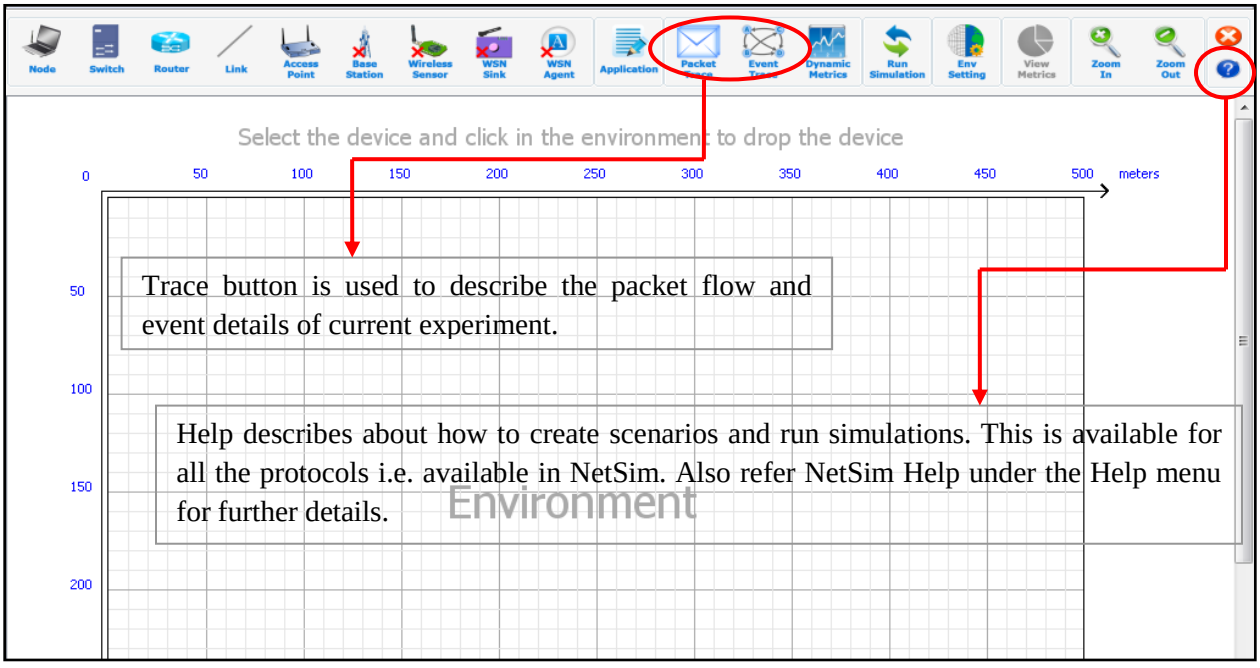
Consists of Frame Capture. Frame Capture is used to capture packets in real time transfer. ***Note: This menu is available only in Academic Version.***

Consists of Animated explanations of networking principles, protocol working and packet formats. ***Note: This menu is available only in Academic Version.***

Simulation

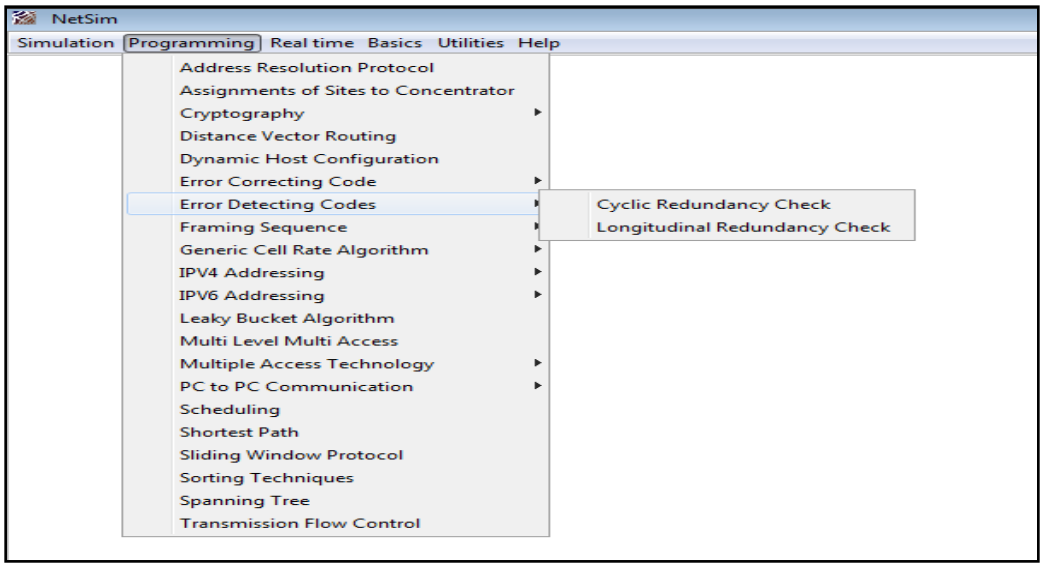
The **Simulation** menu contains options such as **New**, **Open**, **Delete** and **Analytics**. Select the desired protocol in **Internetworks, Legacy, Cellular, BGP, MPLS, Personal Area, Advanced Wireless, Wireless Sensor, Cognitive Radio** and **LTE Networks** under **New** and the screen similar to the one below will open up.

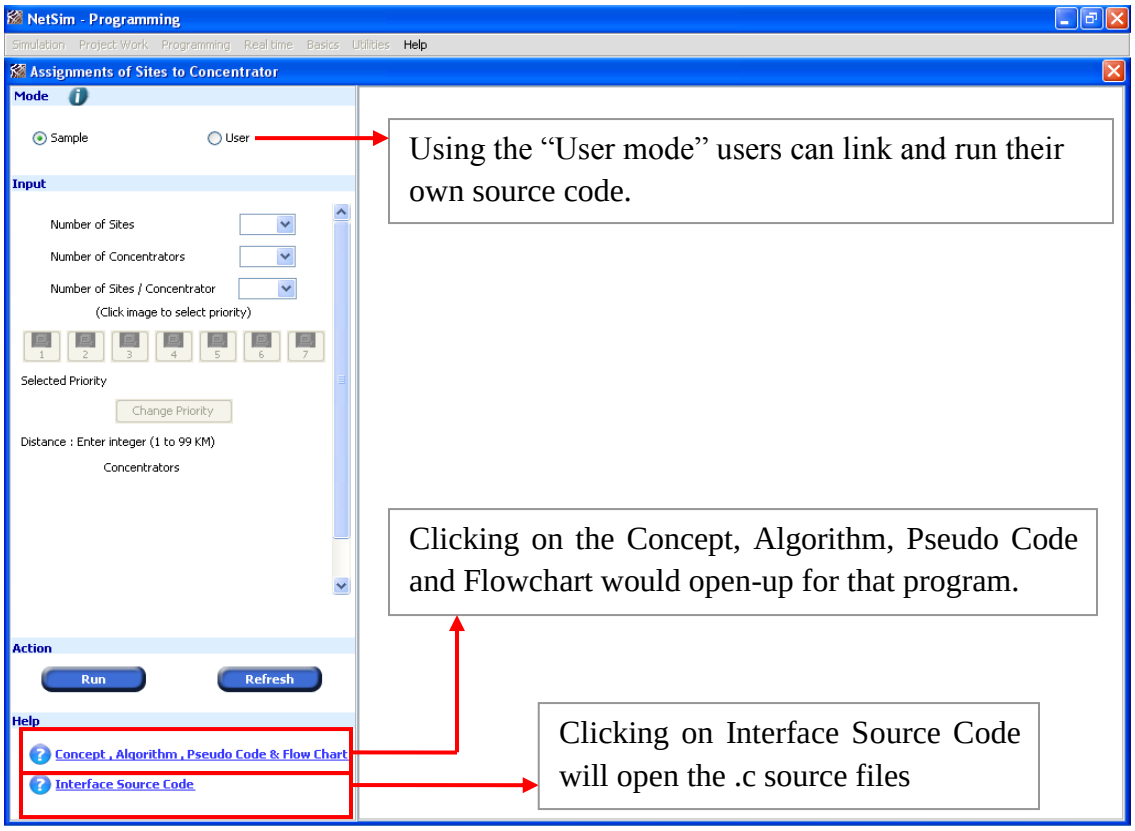



Select the device and click in the environment to drop the device
0
50
100
150
200
250
300
350
400
450
500
meters
Trace button is used to describe the packet flow and event details of current experiment.
Help describes about how to create scenarios and run simulations. This is available for all the protocols i.e. available in NetSim. Also refer NetSim Help under the Help menu for further details.
Environment

Programming

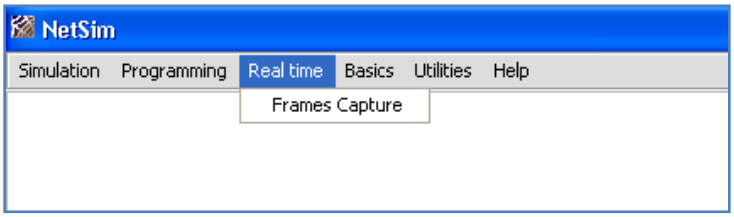
The **Programming** menu contains network programming exercises. Run down the menu and select the desired programming exercise. Upon selection a screen similar to the one shown below will open up. ***Note: This menu is available only in Academic and Standard Version.***




NetSim - Programming
Assignments of Sites to Concentrator
Mode
Sample
User
Input
Number of Sites
Number of Concentrators
Number of Sites / Concentrator
(Click image to select priority)
Selected Priority
Change Priority
Distance : Enter integer (1 to 99 KM)
Concentrators
Action
Run
Refresh
Help
Concept , Algorithm , Pseudo Code & Flow Chart
Interface Source Code
Using the "User mode" users can link and run their own source code.
Clicking on the Concept, Algorithm, Pseudo Code and Flowchart would open-up for that program.
Clicking on Interface Source Code will open the .c source files

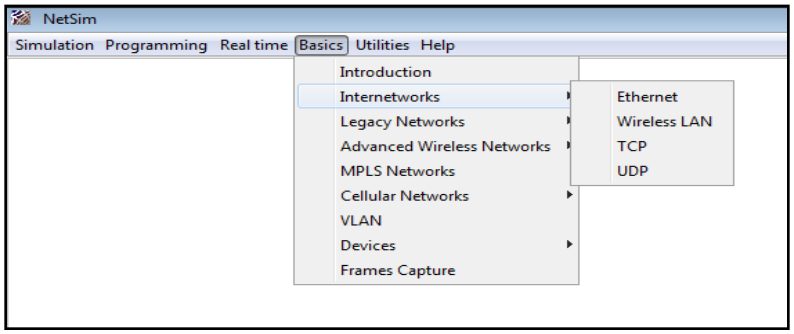
Real time

This section consists of Frame Capture. Frame Capture is used to capture packets in real time transfer. ***Note: This menu is available only in Academic Version.***

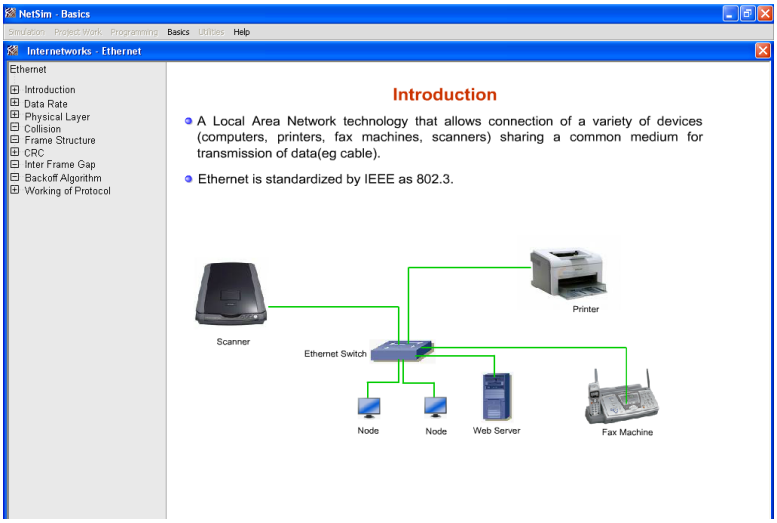


Basics

Select the desired item under **Basics** for animated explanations. ***Note: This menu is available only in Academic Version.***



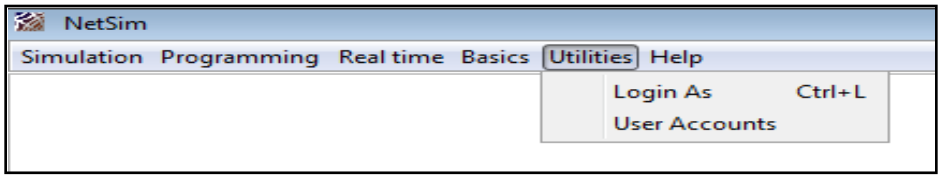
Example: Upon selecting "**Ethernet**" under "**Internetworks**", following screen will open out.



Utilities

This section displays consists of the following options,

- **Login As** - This option is used to login as that user which the user desires. This can be done based on the access provided.
- **User Management** - This option is used for Adding, Deleting, Change Password and Change Mode of the users. Also unwanted experiments can be deleted.

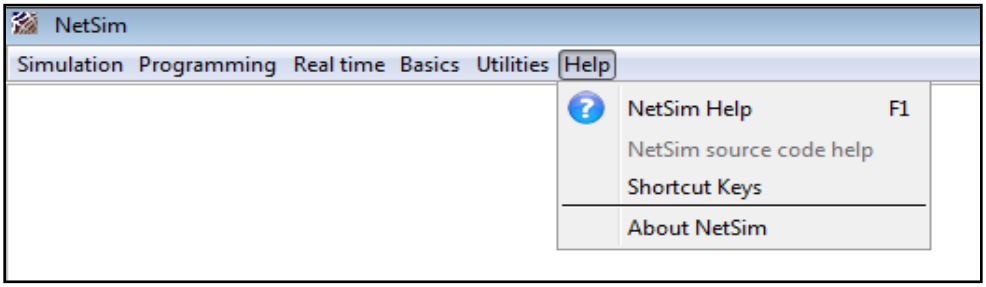


Help

This section displays all the **Help** related to **NetSim**.

NetSim source code help – it contains the documentation about source code *(Available with NetSim Standard as well as Pro Version)*

Shortcut Keys – it contains shortcut keys for all menus and sub menus.



NetSim File Organization

The install directory of NetSim also termed as “application path” is as explained below

Consider *C:\Program Files\NetSim Standard* as <app path>

- *<app path>\bin*
  - ✓ All the “.dll” files like Simulation.dll, jdic.dll, rlm932.dll, etc.
  - ✓ NetSimCore.exe

The following folder contains files essential to developers writing custom code in NetSim.

- *<app path>\src*
  - *<app path>\src\Simulation*(This folder is not available in the Academic version)
    - *<app path>\src\Simulation\Include*
      - ✓ Packet.h
      - ✓ Stack.h
      - ✓ main.h
      - ✓ Protocol.h
      - ✓ List.h
    - *<app path>\src\Simulation\lib*
      - ✓ protocol1.lib
      - ✓ protocol2.lib
      - ✓ protocol3.lib
      - ✓ NetworkStack.lib
      - ✓ Metrics.lib etc.
    - *<app path>\src\Simulation\(Protocol Name)*
      - ✓ .c files
      - ✓ .h files

- *<app path>\src\Programming*
    - ✓ All .c files of programming exercises

Build (as .dll) your custom code into the bin folder of the app path.

- *<app path>\Docs\Sample_Configuration* (This folder contains the sample configuration.xml files) in the corresponding folders \Internetworks, \CognitiveRadio, \MANET,\Zigbee,\WSN, \LTE, \Cellular
- *<app path>\Docs\Help File*
    - Contains NetSim User Manual and Experiment Manual.
    - Contains the help for NetSim source code. (**Note: Not available for Academic version only**)
- *<app path>\Docs\xml*
    - Contains xml files related to NetSim
- *<app path>\Docs\Icons*
    - Contains icons for GUI of NetSim.
- *<app path>\Movie*
    - Contains animations for Basic menu of NetSim. (**Note: Available in Academic version only**).

Running Simulation via Graphical User Interface (GUI)

Internetworks

Internetwork simulation is available in NetSim with LAN-WAN-LAN modeling capability. Internetwork runs Ethernet, Wireless LAN, IP Routing and TCP / UDP and allows users to log packet and event (in NetSim standard version only) traces.

New Experiments

In the **Simulation** menu select → **New** → **Internetworks**

To perform experiments in **Internetworks**, the following steps should be followed,

- **Create Scenario**
- **Connect the devices**
- **Click and drop Application/s to generate traffic**
- **Set Properties for the devices, links and the applications**
- **Enable Packet Trace or/and Event Trace (Optional)**
- **Enable dynamic metrics (Optional)**
- **Run Simulation**

Create Scenario

Adding Devices – Internetworks comes with the palette of various devices like Switch, Router, Wired Node, Wireless Node, AP, etc.

Select the desired devices application icon in the toolbar and click and drop on the environment.

To remove devices or application, right click on the particular icon and then click Remove.

Connect the devices

Select the appropriate link in the toolbar and connect the devices by clicking on the device 1 and device 2.

Set the Properties for the devices, links and applications

- Right click over the devices and then select Properties to set the properties of the links and the devices
- Click and drop Application/s to generate traffic
- Right click over Application and set properties. Multiple applications can be generated by using add button in Application properties.

Enable Packet Trace or/and Event Trace (Optional)

Click Packet Trace icon/ Event Trace icon in the tool bar. Set the name and path and select the required attributes. To get detailed help, please refer Generating Packet trace and Event trace.

Enable Dynamic Metrics (Optional)

Click on dynamic metrics to monitor the value of a parameter during simulation time. To get detailed help, please refer Dynamic Metrics.

Run Simulation - Select the Simulation Time and then Click on "OK" button to start the Simulation

Legacy Networks

New Experiments

In the **Simulation** menu select → **New** → **Legacy Networks**

For example, to arrive **Pure Aloha**,

In the **Simulation** menu select → **New** → **Legacy Networks** → **Pure Aloha**.

To perform experiments in **Legacy Networks**, the following steps should be followed,

- **Create Scenario**
- **Establishing Connections**
- **Set Device Properties**
- **Enable Packet Trace (Optional)**
- **Run Simulation**

Create Scenario

Adding Node:

- **Click** on the **Node** icon in the tool bar and click and drop inside the grid. (*Note*: This is applicable for Pure Aloha and Slotted Aloha)
- Nodes cannot be connected directly to each other because an intermediate connecting component (such as Hub or Concentrator) is required. (Note: This is applicable for Traditional Ethernet, Token Bus and Token Ring)

Adding Hub:

- **Click** on the **Hub** icon in the tool bar and click it onto the environment. By default a Hub has eight ports. (*Note*: This is applicable for Traditional Ethernet and Token Bus)

Adding Concentrator:

- **Click** on the **Concentrator** icon in the tool bar and click it onto the environment. By default a Concentrator consists of eight ports. (*Note*: This is applicable for Token Ring)

Adding CPE (Customer Premise Equipment):

- **Click** on the **CPE** icon in the tool bar, click and drop the CPE on the environment. (*Note*: This is applicable for ATM, X.25 and Frame Relay)

Adding Switch:

- **Click** on the **Switch** icon in the tool bar, click and drop the Switch on the environment. By default a Switch has eight ports.

Establishing Connections

The steps for connecting CPE and Switch are as follows,

- The connections between two **CPE**s cannot be made in the network.
- The connection possibilities are,
    - **CPE** to **Switch** and
    - **Switch** to **Switch**.

(*Note:* Depending upon the simulation, choose either ATM Switch or X.25 Switch or Frame Relay Switch)

Set Node Properties or CPE Properties

Right Click on the appropriate node or CPE and select Properties

Set the Properties for the devices and links

Right click over the devices and then select Properties to set the properties of the links and the devices

Modifying/Viewing/Accepting Properties

On opening an already configured properties of an application the input fields will be frozen (i.e. the input cannot be changed).To modify these values click on the **Modify** button in the screen.

Now the input value can be changed. Click on the **Accept** button, the modified values will be saved.

This **View** button is enabled once the **Accept** Button is clicked. To view the given values, click on the **View** button.

Enable Packet Trace (Optional)

Click Packet Trace icon in the tool bar. Set the name and path and select the required attributes. To get detailed help, please refer Generating Packet trace.

Run Simulation - Select the Simulation Time and then Click on "OK" button to start the Simulation

Advanced Wireless Networks

New Experiments

In the **Simulation** menu select → **New** → **Advanced Wireless Networks**

For example, to arrive **MANET**,

In the **Simulation** menu select → **New** → **Advanced Wireless Networks** →**MANET**

To perform experiments in **Advanced Wireless Networks**, the following steps should be followed,

- **Create Scenario**
- **Set Device Properties**
- **Enable Packet Trace or/and Event Trace (Optional)**
- **Run Simulation**

Create Scenario

Adding Node:

- **Click** on the **Node** icon in the tool bar and click and drop it inside the **grid.** (*Note*: A Node cannot be placed on another Node. A Node cannot float outside of the grid.)

Adding Base Station and Subscriber (*Note*: This is applicable for Fixed Wi-MAX)

- **Click** on the **Base Station** icon in the tool bar and **click** it onto the environment.
- **Click** on the **Subscriber** icon in the tool bar and **click** and **drop** it on the **Base Station** coverage area.

- **Subscriber** cannot be placed on another **Subscriber**. It has to be clicked and placed on the **Base Station** coverage area.

Set Node Properties

- **Right click** on the appropriate node to select Properties. (*Note*: This is applicable for MANET)

Set Base Station (BS) Properties

- **Right click** on the **Base Station (BS)** and set **Properties**. (*Note*: This is applicable for Fixed Wi-MAX)

Set Subscriber Properties

- **Right Click** on the appropriate Subscriber to select Properties. (*Note*: This is applicable for Fixed Wi-MAX)

Set Environment Properties

- **Right click** on the Environment and **click Properties**. (*Note*: This is applicable for MANET)

Modifying/Viewing/Accepting Properties

On opening an already configured properties of an application the input fields will be frozen (i.e. the input cannot be changed).To modify these values click on the **Modify** button in the screen. Now the input value can be changed. Click on the **Accept** button, the modified values will be saved.

This **View** button is enabled once the **Accept** Button is clicked. To view the given values, click on the **View** button.

Enable Packet Trace or/and Event Trace (Optional)

Click Packet Trace icon/ Event Trace icon in the tool bar. Set the name and path and select the required attributes. To get detailed help, check Generating Packet trace and Event trace.
(*Note*: Event Trace is applicable only for MANET)

Run Simulation - Select the Simulation Time and then Click on “OK” button to start the Simulation

BGP Networks

New Experiments

In the **Simulation** menu select **Simulation → New → BGP Networks**

To perform experiments in **BGP Networks**, the following steps should be followed,

- **Create Scenario**
- **Establishing Connections**
- **Set Device Properties**
- **Set Link Properties**
- **Click and drop applications and configure**
- **Enable Packet Trace or/and Event Trace (Optional)**
- **Enable Dynamic Metrics (Optional)**
- **Run Simulation**

Create Scenario

Adding Border Router:

- **Click and drop** the **Border Router** icon from the tool bar. (*Note*: Maximum you can have 3 Autonomous systems in a single scenario.)

Adding Internal Router:

- **Click** on the **Internal Router** icon in the tool bar and drop the Internal Router onto the Autonomous systems created. By default a Router has eight ports.

Establishing Connections

The steps for connecting devices in BGP networks are as follows,

- The connections between two **wired nodes** cannot be made in the network.
- The connection possibilities are
    - **Wired Node** to **Internal Router**
    - **Internal Router** to **Border Router**
    - **Border Router** to **Border Router**

Set the Properties for the devices, links and applications

- Right click over the devices and then select Properties to set the properties of the links and the devices

- Click and drop Application/s to generate traffic
- Right click over Application and set properties. Multiple applications can be generated by using add button in Application properties.

Modifying/Viewing/Accepting Properties

On opening an already configured properties of an application the input fields will be frozen (i.e. the input cannot be changed).To modify these values click on the **Modify** button in the screen. Now the input value can be changed. Click on the **Accept** button, the modified values will be saved.

This **View** button is enabled once the **Accept** Button is clicked. To view the given values, click on the **View** button.

Enable Packet Trace or/and Event Trace (Optional)

Click Packet Trace icon/ Event Trace icon in the tool bar. Set the name and path and select the required attributes. To get detailed help, please refer Generating Packet trace and Event trace.

Enable Dynamic Metrics (Optional)

Click on dynamic metrics to monitor the value of a parameter during simulation time. To get detailed help, please refer Dynamic Metrics.

Run Simulation - Select the Simulation Time and then Click on "OK" button to start the Simulation.

MPLS Networks

New Experiments

In the **Simulation** menu select **Simulation → New → MPLS Networks**

To perform experiments in **MPLS Networks**, the following steps should be followed,

- **Create Scenario**
- **Establishing Connections**
- **Set Device Properties**
- **Set Link Properties**

- **Enable Packet Trace (Optional)**
- **Run Simulation**

Create Scenario

Adding CPE - Click on the **CPE** icon in the toolbar and drop the CPE on the **environment**.

Adding Router - Click on the **Router** icon in the tool bar**,** click and drop the Router on the environment. By default a Router has eight ports.

Establishing Connections

The steps for connecting CPE and Router are as follows:

- The connections between two **CPEs** cannot be made in the network.
- The connection possibilities are
    - **CPE** to **Router** and
    - **Router** to **Router**.

Set the Properties for the devices and links

Right click over the devices and then select Properties to set the properties of the links and the devices

Modifying/Viewing/Accepting Properties

On opening an already configured properties of an application the input fields will be frozen (i.e. the input cannot be changed).To modify these values click on the **Modify** button in the screen. Now the input value can be changed. Click on the **Accept** button, the modified values will be saved.

This **View** button is enabled once the **Accept** Button is clicked. To view the given values, click on the **View** button.

Enable Packet Trace (Optional)

Click Packet Trace icon in the tool bar. Set the name and path and select the required attributes. To get detailed help, please refer Generating Packet trace.

Run Simulation - Select the Simulation Time and then Click on “OK” button to start the Simulation

Cellular Networks

New Experiments

In the **Simulation** menu select → **New** → **Cellular Networks**

For Example, to arrive **CDMA**

In the **Simulation** menu select → **New** → **Cellular Networks** → **CDMA**

To perform experiments in **Cellular Networks (GSM/CDMA)**, the following steps should be followed,

- **Create Scenario**
- **Set Device Properties**
- **Set link properties**
- **Click and drop applications and configure**
- **Enable Packet Trace or/and Event Trace (Optional)**
- **Enable Dynamic Metrics (Optional)**
- **Run Simulation**

Create Scenario

Adding Base Transceiver Station (BTS) - Click on the **BTS** icon in the toolbar and click it onto the **environment**.

Adding Mobile Switching Centre (MSC) - Click and drop **MSC** in the environment.

Adding Mobile Station (MS) -

- **Click** on the **Mobile Station** icon in the tool bar, click and drop it on the **Base Station coverage** area.
- **Mobile Station** cannot be placed on another **Mobile Station**. It has to be clicked and placed on the **Base Station** coverage area.

Set the Properties for the devices, links and applications

- Right click over the devices and then select Properties to set the properties of the links and the devices
- Click and drop Application/s to generate traffic
- Right click over Application and set properties. Multiple applications can be generated by using add button in Application properties.

Enable Packet Trace or/and Event Trace (Optional)

Click Packet Trace icon/ Event Trace icon in the tool bar. Set the name and path and select the required attributes. To get detailed help, please refer Generating Packet trace and Event trace.

Enable Dynamic Metrics (Optional)

Click on dynamic metrics to monitor the value of a parameter during simulation time. To get detailed help, please refer Dynamic Metrics.

Run Simulation - Select the Simulation Time and then Click on "OK" button to start the Simulation.

Wireless Sensor Networks

New Experiments

In the **Simulation** menu select **Simulation → New → Wireless Sensor Networks**

To perform experiments in **Wireless Sensor Networks**, the following steps should be followed,

- **Create Scenario**
- **Set Device Properties**
- **Set Environment Properties**
- **Enable Packet Trace or/and Event Trace(Optional)**
- **Enable Dynamic Metrics (Optional)**
- **Run Simulation**

Create Scenario

Adding Sink Node- Click on the **Sink Node** icon in the toolbar and click and drop inside the grid.

Adding Sensor - Click on the **Sensor Node** icon in the toolbar and click and drop inside the grid.

Adding Agent- Click on the **Agent** icon in the toolbar and click and drop inside the grid.

Set the Properties for the devices

Right click over the devices (sensor, sink node and agent) and then select Properties to set the properties of the devices.

Enable Packet Trace or/and Event Trace (Optional)

Click Packet Trace icon/ Event Trace icon in the tool bar. Set the name and path and select the required attributes. To get detailed help, please refer Generating Packet trace and Event trace.

Enable Dynamic Metrics (Optional)

Click on dynamic metrics to monitor the value of a parameter during simulation time. To get detailed help, please refer Dynamic Metrics.

Run Simulation - Select the Simulation Time and then Click on "OK" button to start the Simulation.

Personal Area Networks

New Experiments

In the **Simulation** menu select **Simulation → New → Personal Area Networks**

For example, to arrive **ZigBee**

In the **Simulation** menu select **Simulation → New → Personal Area Networks → ZigBee**

To perform experiments in **Personal Area Networks**, the following steps should be followed,

- **Create Scenario**
- **Set Device Properties**
- **Set Environment Properties**
- **Click and drop applications and configure**
- **Enable Packet Trace or/and Event Trace (Optional)**
- **Enable Dynamic Metrics (Optional)**
- **Run Simulation**

Create Scenario

Adding Node -

- **Click** on the **Node** icon in the toolbar and click and drop it inside the grid (i.e. **Visibility Range** - The systems can move and communicate in this range only).
- A Node cannot be placed on another Node. A Node cannot float outside the **grid**.

Adding PAN Coordinator - Click on the **PAN Coordinator** icon in the toolbar and click and drop inside the **grid**.

Set Environment Properties

Right click in side of the on the Environment and **click Properties.**

Set the Properties for the devices, links and applications

- Right click over the devices and then select Properties to set the properties of the links and the devices
- Click and drop Application/s to generate traffic
- Right click over Application and set properties. Multiple applications can be generated by using add button in Application properties.

Modifying/Viewing/Accepting Properties

On opening an already configured properties of an application the input fields will be frozen (i.e. the input cannot be changed).To modify these values click on the **Modify** button in the screen. Now the input value can be changed. Click on the **Accept** button, the modified values will be saved.

This **View** button is enabled once the **Accept** Button is clicked. To view the given values, click on the **View** button.

Enable Packet Trace or/and Event Trace (Optional)

Click Packet Trace icon/ Event Trace icon in the tool bar. Set the name and path and select the required attributes. To get detailed help, please refer Generating Packet trace and Event trace.

Enable Dynamic Metrics (Optional)

Click on dynamic metrics to monitor the value of a parameter during simulation time. To get detailed help, please refer Dynamic Metrics.

Run Simulation - Select the Simulation Time and then Click on “OK” button to start the Simulation.

Cognitive Radio Networks

Cognitive Radio Network simulation is available from v7 of NetSim. Cognitive Radio Networks allows you to connect, if required, with Ethernet, Wireless LAN, IP Routing, TCP / UDP and allows users to log packet and event (in NetSim standard version only) traces.

New Experiments

In the **Simulation** menu select → **New** → **Cognitive Radio Networks.**

Then Select either Map View or Grid view, and to perform experiments in **Cognitive Radio Networks** the following steps should be followed:

- **Create Scenario**
- **Connect the devices**
- **Set Properties for the devices and the links**
- **Click and drop applications and configure**
- **Enable Packet Trace or/and Event Trace (Optional)**
- **Enable Dynamic Metrics (Optional)**
- **Run Simulation**

Create Scenario

Adding Devices –

- Cognitive Radio Networks comes with the palette of various devices like, Cognitive Radio CPE, Cognitive Radio Base Station, Switch, Router, Wired Node, Wireless Node, Access point etc.
- Select the desired devices in the toolbar and click and drop on the environment.
- To remove devices, right click on the particular device and then click Remove.

Connect the devices

Select the appropriate link in the toolbar and connect the devices by clicking on the device 1 and device 2.

Set the Properties for the devices, links and applications

- Right click over the devices and then select Properties to set the properties of the links and the devices
- Click and drop Application/s to generate traffic

- Right click over Application and set properties. Multiple applications can be generated by using add button in Application properties.

Enable Packet Trace or/and Event Trace (Optional)

Click Packet Trace icon/ Event Trace icon in the tool bar. Set the name and path and select the required attributes. To get detailed help, please refer Generating Packet trace and Event trace.

Enable Dynamic Metrics (Optional)

Click on dynamic metrics to monitor the value of a parameter during simulation time. To get detailed help, please refer Dynamic Metrics.

Run Simulation - Select the Simulation Time and then Click on "OK" button to start the Simulation.

Notes on Cognitive Radio implementation in NetSim:

- CR BS allocates max one symbol per CPE. If the generation rate is more than the data filled in one symbol then allocation fails and it results in zero throughput.
- The first symbol is reserved for CR control frames or any broadcast PDU
- NetSim gives App layer throughput which is lesser than that of MAC layer throughput because of
  - TCP connection establishment
  - ARP set-up
  - Service flow creation CPE-BS and BS-CPE
  - BW request
- To avoid the above effects
  - Generate custom traffic
  - Set DL/UL Ratio as 1:1 so that the BS transmits whatever it receives
  - Run UDP in transport layer
  - Use static ARP
  - Run Simulation for more than 100 sec

Long Term Evolution

LTE simulation per 3GPP.36 standard is available from version 8 of NetSim

New Experiments

In the **Simulation** menu select→**New**→**LTE Networks.** Then the following steps should be performed:

- **Create Scenario**
- **Connect the devices**
- **Set Properties for the devices and the links**
- **Click and drop applications and configure**
- **Enable Packet Trace or/and Event Trace (Optional)**
- **Enable Dynamic Metrics (Optional)**
- **Run Simulation**

Create Scenario

Adding MME- Click on the **Router** icon in the tool bar, click and drop the **MME** (**Mobility Management Entity)** onto the **environment.**

Adding ENB - Click on the **Evolved node B (ENB)** icon in the toolbar and click it onto the **environment**.

Adding UE –

- **Click** on the **UE (User Equipment)** icon from the **Node** icon in the toolbar, click and drop it on the **ENB coverage** area.
- **UE** cannot be placed on another **UE**. It has to be clicked and placed on the **ENB node coverage** area.

Connect the devices

Select the appropriate link in the toolbar and connect the devices by clicking on the device 1 and device 2.

Set the Properties for the devices, links and applications

- Right click over the devices and then select Properties to set the properties of the links and the devices
- Click and drop Application/s to generate traffic

- Right click over Application and set properties. Multiple applications can be generated by using add button in Application properties.

Enable Packet Trace or/and Event Trace (Optional)

Click Packet Trace icon/ Event Trace icon in the tool bar. Set the name and path and select the required attributes. To get detailed help, please refer Generating Packet trace and Event trace.

Enable Dynamic Metrics (Optional)

Click on dynamic metrics to monitor the value of a parameter during simulation time. To get detailed help, please refer Dynamic Metrics.

Run Simulation - Select the Simulation Time and then Click on "OK" button to start the Simulation.

IP Addressing in NetSim

When you create a network using the GUI, NetSim will automatically configure the IP address of the devices in the scenario.

Consider the following scenarios:

If you create a network with two wired nodes and a switch, the IP addresses are assigned as 10.0.1.2 and 10.0.1.3 for the two wired nodes. The default subnet mask is assigned to be 255.255.0.0. It can be edited to 255.0.0.0 (Class A) or 255.255.255.0 (Class C) subnet masks. Both the nodes are in the same network (10.0.0.0).

Similarly, if you create a network with a router and two wired nodes, the IP addressed are assigned as 11.1.1.2 and 11.2.1.2 for the two wired nodes. The subnet mask is default as in above case, i.e., 255.255.0.0. The IP address of the router is 11.1.1.1 and 11.2.1.1 respectively for the two interfaces. Both the nodes are in different networks (11.1.0.0 and 11.2.0.0) in this case.

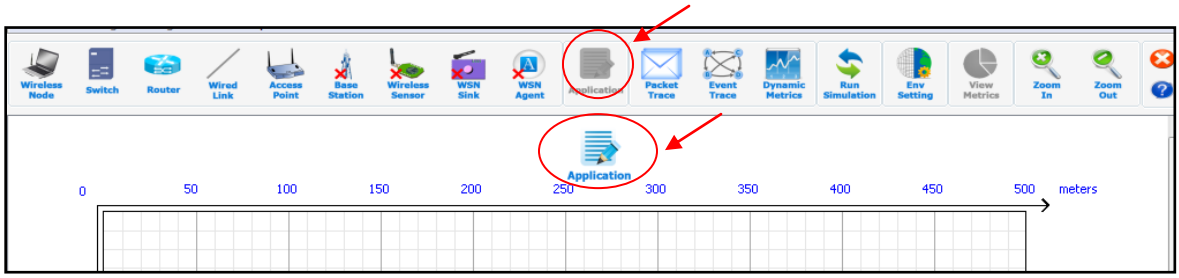
The same logic is extended as the number of devices is increased.

Applications

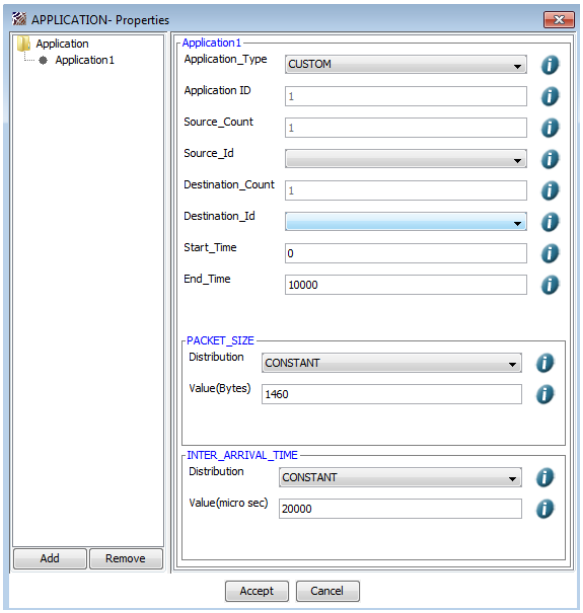
NetSim allows users to model and simulate the applications:

1. Custom
2. Database
3. FTP
4. Email
5. HTTP
6. PEER_TO_PEER
7. Video
8. Voice

To set up the application click and drop the application icon from the tool bar as shown below.



Right click on the application icon and select properties



This properties window allows you to set the traffic. You can add (or) delete one or more applications.

Common properties for all the traffic types

Application ID

This property represents the unique identification number of the application.

Start time

This property represents the start time of the application in seconds.

End time

This property represent the end time of the application in seconds.

Note: *Suppose Start time is 1 and end time is 10 then application starts generating traffic at 1st second and ends at 10th second.*

Source Count

This property represents number of sources for the application. Voice, Video, FTP, Database and Custom applications have only one source.

Source ID

This property represents the unique identification number of the source.

Destination Count

This property represents number of destinations for the application. Voice, Video, FTP, Database and Custom applications have only one destination.

Destination ID

This property represents the unique identification numbers of the destination.

Custom

Packet Size

Distribution: The options available for distribution are,

- **Constant**
- **Exponential**

Packet Size (Bytes): Sets the size of the packets being generated by the chosen distribution. By default 1460 bytes is entered.

Inter Arrival Time

This indicates the time gap between packets.

Distribution: The options available for distribution are,

- **Constant**
- **Exponential**

Inter Arrival Time: Enter the average inter-arrival time between packets. A lower inter-arrival time would lead to a higher generation rate and the vice versa. By default 20000 Micro Sec is entered.

Voice

Codec

Codec is the component of any voice system that translates between analog speech and the bits used to transmit them. Every codec transmits a burst of data in a packet that can be reconstructed into voice. Five different standards of voice codec's are given which can be selected depending on the variations required.

Service Type

- **CBR** - CBR stands for Constant Bit Rate. Packets of constant size are generated at constant inter arrival times.
- **VBR** - VBR stands for Variable Bit Rate. The two types of Suppression Model that can be selected are,
    - **Deterministic**
    - **Markov Chain**

Video

Video is an electronic medium for the recording, copying and broadcasting of moving visual images.

Model Type

- **Continuous Normal VBR** – This model is the simplest of all models. It uses Normal Distribution for the generation of bits per pixel. In this model, consecutive packet sizes are independent of each other.
    - **Frames per second** – Number of frames arriving per second. This is in the range of 10 – 50.
    - **Pixels per frame** -Number of pixelsin each frame. This is in the range of 10000 – 100000.
    - **Bits per pixel (μ)**– Mean value of the normal distribution used to generate the value of bits per pixel.
    - **Bits per pixel (Σ)** – Standard Deviation of the normal distribution used to generate the value of bits per pixel.
- **Continuous State Autoregressive Markov** –This model incorporates the autocorrelation between the frames. Also, current packet size depends on the previous packet size via the first order autoregressive Markov process.
    - **Frames per second** – Number of frames arriving per second. This is in the range of 10 – 50.
    - **Pixels per frame** - Number of pixels in each frame. This is in the range of 10000 – 100000.
    - **Constant A**– First order autoregressive Markov processλ(n) can be generated by the recursive relation λ(n)=aλ(n-1)+bw(n).
    - **Constant B**– First order autoregressive Markov process λ(n) can be generated by the recursive relation λ(n)=a λ(n-1)+bw(n).
    - **Eta**– The steady-state average E(λ)and discreet auto covariance C(n) are given byE(λ) = (b / (1-a)) η C(n)=(b2/(1-a2))an where η is the Gaussian parameter.
- **Quantized State Continuous Time Markov** –In this model the bit rate is quantized into finite discreet levels. This model takes uniform quantization step as A bits/pixel. There are M + 1 possible levels(0, A, ….., MA).Transitions between levels are assumed to occur with exponential rates that may depend on the current level. This

model is approximating the bit rate by a continuous time process λ(t) with discreet jumps at random Poisson time.

- **Frames per second** – Number of frames arriving per second. This is in the range of 10 – 50.
- **Pixels per frame** - Number of pixels in each frame. This is in the range of 10000 – 100000.
- **No of Multiplexed Sources**– This model considers the aggregate instantaneous input rate λN(t) instead of the single source bit rate λ(t). The total rate is the sum of N independent random processes each with mean E(λ) and variance C(0) at steady state.Therefore, the steady state-state mean of λN(t) will be E(λ N)=N x E(λ) bits/pixel.
- **Quantization Level**– This model takes uniform quantization step as A bits/pixel. There are M + 1 possible levels (0, A, ….., MA). Transitions between levels are assumed to occur with exponential rates that may depend on the current level.

- **Simple IPB Composite Model**–In this model, the frames are organized as IBBPBBPBBPBBIBBPBB… i.e., 12 frames in a Group of Pictures(GOP).Generate X0 from a Gaussian distribution N(0, y 0).Set initial value N0= 0, D0 = 1.

  For k = 1, 2,…, N-1, calculate Φkj , j = 1, 2,…,k iteratively using the following formulae

  Nk = r(k) – j=1Σk-1Φk-1,j r(k-j)

  Dk = Dk-1 –(N2k-1/Dk-1)

  Φkk = Nk / Dk

  Φkj = Φk-1, j-ΦkkΦk-1,k-j j=1,….,k-1

  mk = j = 1ΣkΦkjXk-j

  y k = (1- Φ2kk) yk-1

Finally, eachXk is chosen from N(mk, y k). Thus we get a process X with ACF approximating to r(k).

The auto correlation function can be calculated in a recursive way as follows:

r(0) = 1, r(k+1) = ((k+d)/(k+1))r(k)

where d= H-0.5.

H is called the Hurst parameter k-β is used as the ACF of a self-similar process.We get the value of H parameter for a self-similar process using the relationship,

B = 2 – 2H

Distribution of these data is Gaussian. For data to be Beta distributed, the following mapping is being used

Yk = F-1β (FN(Xk)), k>0

Xk :Self-similar Gaussian process,

FN :The cumulative probability of normal distribution,

F-1β: The inverse cumulative probability functions of the Beta model.

- **Frames per second** – Number of frames arriving per second. This is in the range of 10 – 50.
- **Hurst I**–Refer i-button help of Simple IPB Composite Model.
- **Hurst B**– Refer i-button help of Simple IPB Composite Model.
- **Hurst P**– Refer i-button help of Simple IPB Composite Model.
- **Gamma I** – Refer i-button help of Simple IPB Composite Model.
- **Gamma B**– Refer i-button help of Simple IPB Composite Model.
- **Gamma P**– Refer i-button help of Simple IPB Composite Model.
- **Eta I** – Refer i-button help of Simple IPB Composite Model.
- **Eta B** – Refer i-button help of Simple IPB Composite Model.
- **Eta P**– Refer i-button help of Simple IPB Composite Model.

FTP

File Size

Distribution: The options available for distribution are,

- **Constant**
- **Exponential**

File Size (Bytes): Sets the size of the packets being generated by the chosen distribution. By default 1000000 bytes is entered.

File_Inter Arrival Time

This indicates the time gap between packets.

Distribution: The options available for distribution are,

- **Constant**
- **Exponential**

Inter Arrival Time: Enter the average inter-arrival time between packets. A lower inter-arrival time would lead to a higher generation rate and the vice versa. By default 5 Sec is entered.

Database

Transaction Size

Distribution: The options available for distribution are,

- **Constant**
- **Exponential**

Packet_Size (Bytes): Sets the size of the packets being generated by the chosen distribution. By default 10000 bytes is entered.

Transaction Inter Arrival Time

This indicates the time gap between packets.

Distribution: The options available for distribution are,

- **Constant**
- **Exponential**

Inter Arrival Time: Enter the average inter-arrival time between packets. A lower inter-arrival time would lead to a higher generation rate and the vice versa. By default 1000000 Micro Sec is entered.

Peer to Peer

File Size

Distribution: The options available for distribution are,

- **Constant**
- **Exponential**

Value (Bytes): This represents the size of the file in bytes.

Piece size (Bytes): Each file is divided into equal sized pieces and then the piece of the data is transmitted. This property represents the size of each piece.

HTTP

Http_request_interarrival_time

This indicates the time gap between the pages.

Distribution: The options available for distribution are,

- **Constant**
- **Exponential**

Inter Arrival Time (micro sec): This represents the rate at which client sends the requests.

Page_property

Distribution: The options available for distribution are,

- **Constant**
- **Exponential**

Page Count: This represents the number of pages received from the server.

Page Size (Bytes): This represents the size of each page that is received from the server.

Email

Email_Receive

This represents the rate at which emails are received.

Duration_Distribution: The options available for distribution are,

- **Constant**
- **Exponential**

Duration: Time gap between two successive receiving emails in seconds.

Email_Size_Distribution: The options available for distribution are,

- **Constant**
- **Exponential**

Email_Size: It represents the size of the email that is received.

Email_Send

This represents the rate at which emails are sent.

Duration_Distribution: The options available for distribution are,

- **Constant**
- **Exponential**

Duration: Time gap between two successive sending emails in seconds.

Email_Size_Distribution: The options available for distribution are,

- **Constant**
- **Exponential**

Email_Size: It represents the size of the email that is sent.

Priority and QoS of Applications

The various traffics generated in NetSim have the following priority and QoS values:

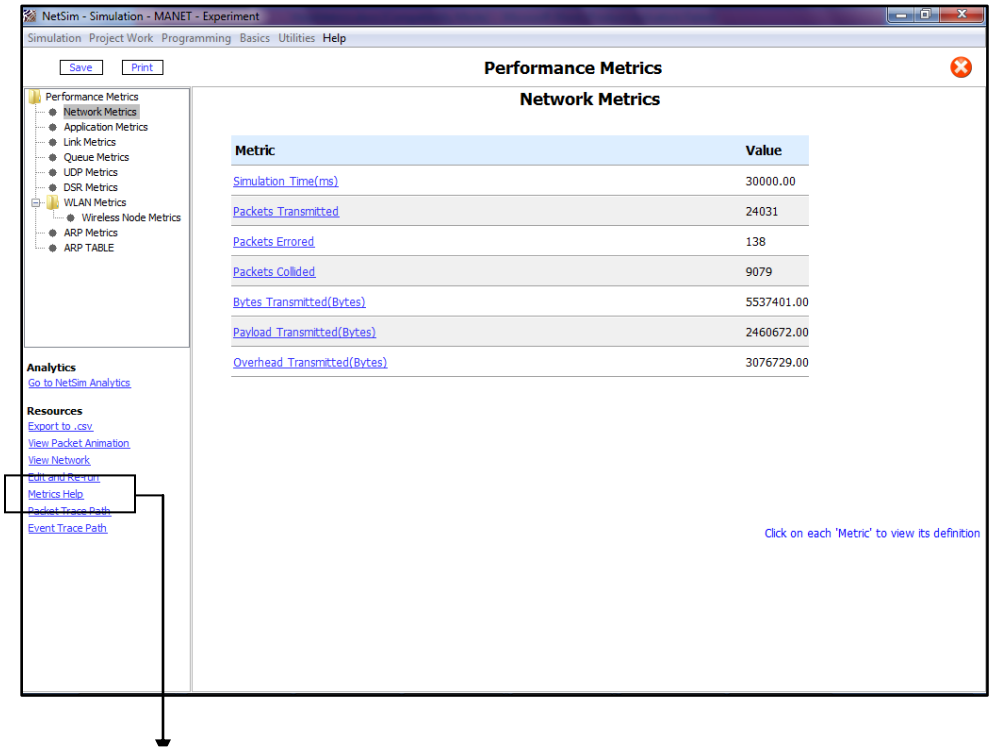
Application Type	Priority Value	Priority	QoS Class
Voice – One way	8	High	RTPS
Voice – Two way	8	High	UGS
Video	6	Medium	nRTPS
FTP	2	Low	BE
Database	2	Low	BE
Custom	2	Low	BE

Note: Priority of "Normal" has a Priority Value of 4 and "nRTPS" QoS Class. Ex: Video over TCP.

Performance Metrics

NetSim provides distinct quantitative metrics at various abstraction levels such as Network Metrics, Link Metrics, TCP Metrics, Application Metrics, etc at the end of simulation. With the help of metrics, users can analyze the behavior of the modeled network and can compare the impact of different algorithms on end-to-end behavior.

After simulation of a scenario is performed, the NetSim Performance Metrics are shown on the screen as shown below



Click on Metrics Help to view the definition of each metric

Note about metrics: The metrics are calculated at each layer and might not be equivalent to the same metric calculated at a different layer. For exactness and precision we recommend users also verify the results with the event trace & packet trace generated by NetSim.

Note about packet transmission: The Network Stack forms the core of NetSim's architecture. The Stack consists of five IN and OUT events: PHYSICAL_IN, MAC_IN, NETWORK_IN, TRANSPORT_IN, APPLICATION_IN and APPLICATION_OUT,TRANSPORT_OUT, NETWORK_OUT, MAC_OUT, PHYSICAL_OUT. All the packets when transferred between devices go through the above events in order. IN events occur when the packet is entering a device and all the OUT events occur when packet leaves a device.
The following table lists the various files that will be written in the NetSim install directory/ IO path on completion of simulation.

S. No	File	Contents
1	Metrics.txt	Contains the metrics of the network that is simulated recently .
2	MetricsGraph.txt	Contains the data required to plot the graph between the various parameters in metrics against time
3	LicenseErrorLog.txt	Contains the status of the communication between the NetSim dongle and the client.
4	ConfigLog.txt	This file will be written while reading the Configuration file. Provides errors if there are errors in the configuration file.
5	LogFile.txt	Contains the logs as the control flows across various layers in the Network Stack
6	PacketTrace.txt	Contains the detailed packet information. This file will be written only when Packet Trace is enabled.
7	EventTrace.txt	Contains the information about each event. This file will be written only when Event Trace is enabled.

If NetSim runs via the UI, then the metrics will be displayed automatically at the end of simulation with illustrative pie charts and tables.

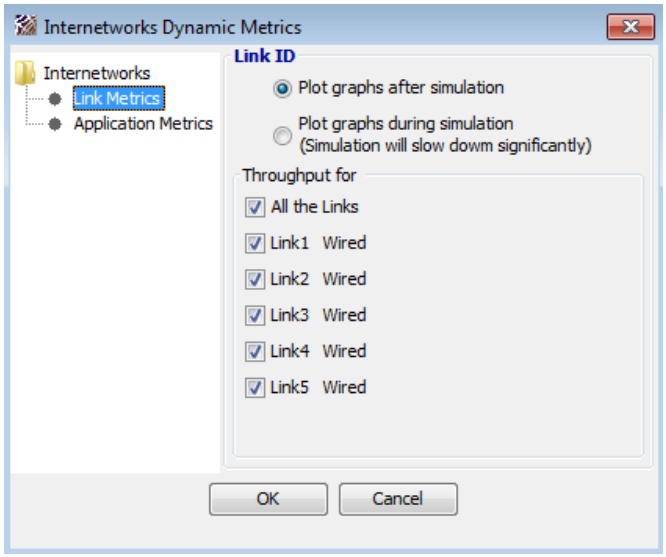
If NetSim runs via CLI, then the metrics will be written into Metrics.txt and MetricsGraph.txt.

Dynamic Metrics (Available in Standard and Pro Version only)

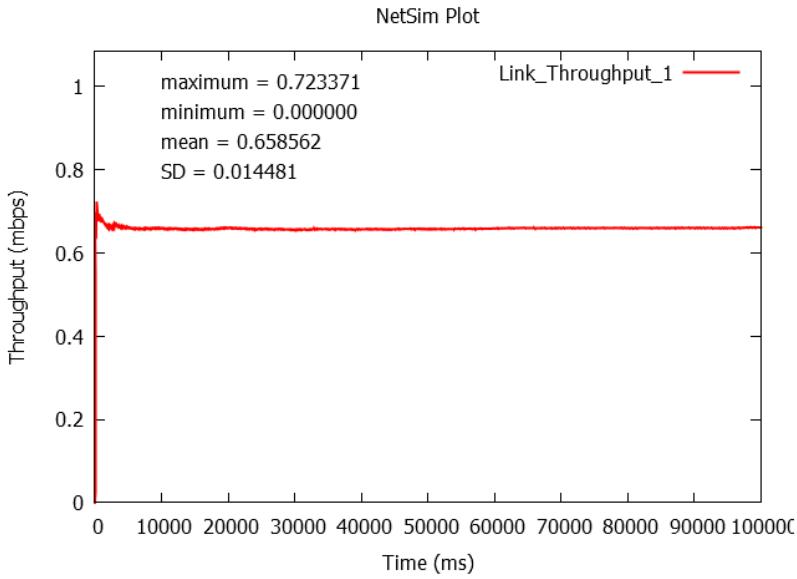
Dynamic metrics allows users to monitor the value of a parameter over simulation time.

As of NetSim v8, link and application throughputs can be dynamically plotted. These plots can be done offline or during simulation. Note that enabling them to be plotted during simulation would consume a lot of memory and slow down the simulation significantly.

Dynamic metrics can be set by clicking on the dynamic metrics icon in the GUI, and selecting those metrics that you wish to monitor.



Upon completion of the simulation dynamic metrics plots can be viewed by clicking on Link throughput in Metrics screen.



Generating Packet Trace (Available in Standard and Pro Version only)

Introduction:

NetSim allows users to generate trace files which provide detailed packet information useful for performance validation, statistical analysis and custom code de-bugging. Packet Trace logs a set of chosen parameters for every packet as it flows through the network such as arrival times, queuing times, departure times, payload, overhead, errors, collisions etc.

By providing a host of information and parameters of every packet that flows through the network, packet trace provides necessary forensics for users to catch logical errors without setting a lot of breakpoints or restarting the program often. Window size variation in TCP, Route Table Formation in OSPF, Medium Access in Wi-fi, etc, are examples of protocol functionalities that can be easily understood from the trace.

> ***Note:*** *Turning on Packet Trace will slow down the simulation significantly*

By default, the packet tracing option is turned off.

How to enable Packet Trace via UI?

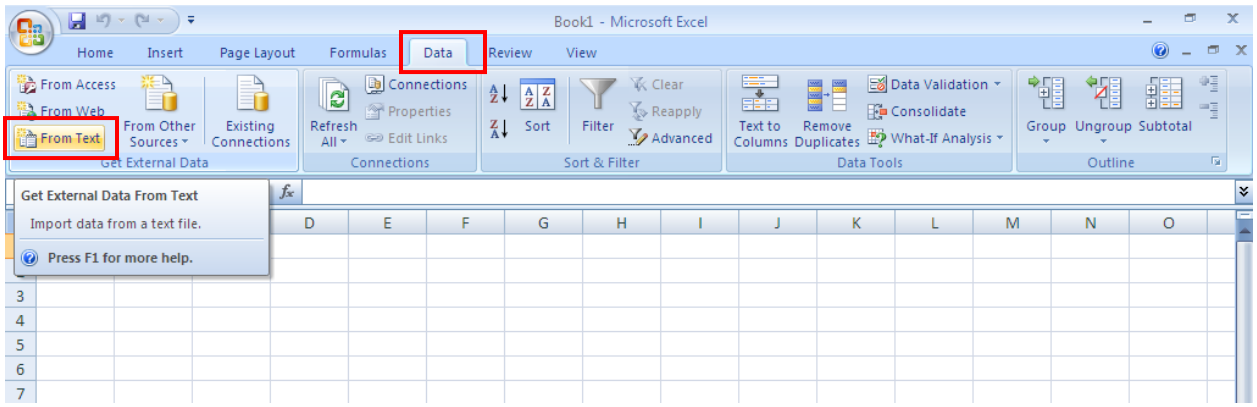
If NetSim runs via the UI, packet trace can be turned on by clicking the Packet Trace icon in the tool bar and selecting the required fields in the packet trace. NetSim will write the packet trace to the specified path during simulation.

How to enable Packet Trace via CLI?

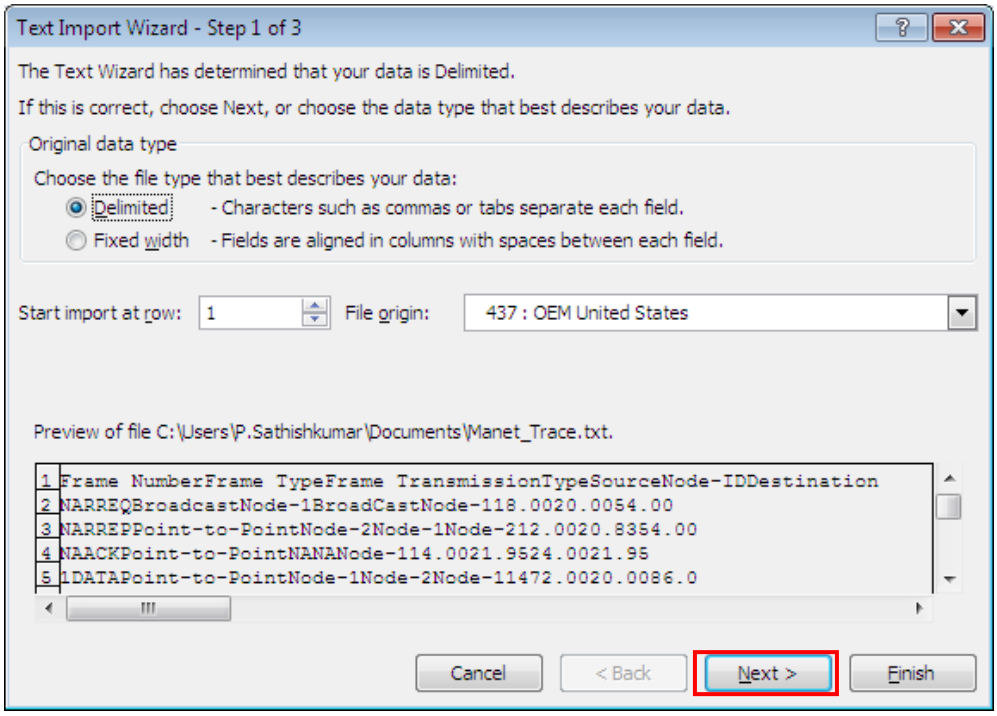
If NetSim runs via CLI, then the packet trace can be turned on by enabling the packet trace in the STATISTICS_COLLECTION tag of the configuration file. NetSim will write the packet trace to the specified path during simulation.

How to import Packet Trace to Excel?

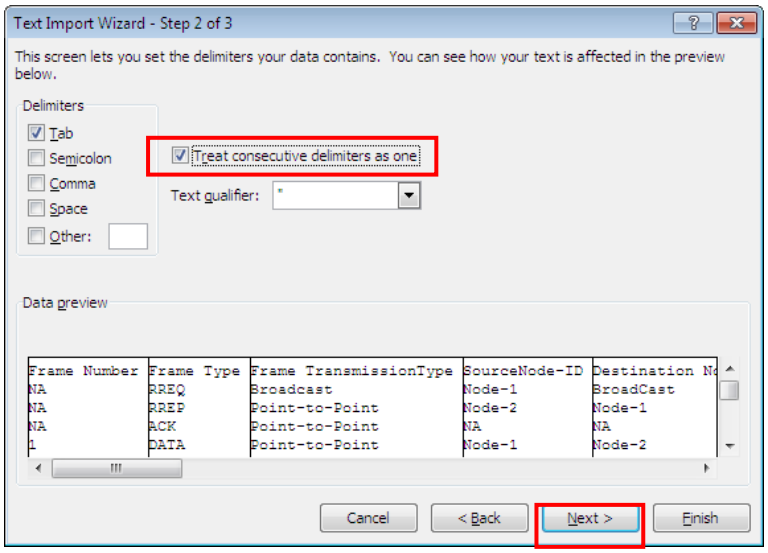
Step 1: Open Excel sheet, select Data → From Text, it will ask for file, then select the trace file which you want to export to excel sheet.

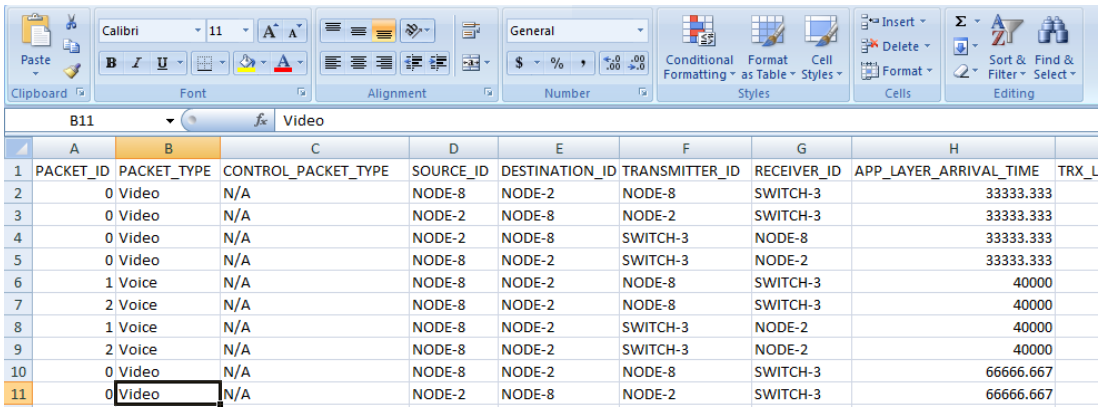


Step 2: Select the file click import, the following window will appear,


Text Import Wizard - Step 1 of 3
The Text Wizard has determined that your data is Delimited.
If this is correct, choose Next, or choose the data type that best describes your data.
Original data type
Choose the file type that best describes your data:
Delimited - Characters such as commas or tabs separate each field.
Fixed width - Fields are aligned in columns with spaces between each field.
Start import at row: 1
File origin: 437 : OEM United States
Preview of file C:\Users\P.Sathishkumar\Documents\Manet_Trace.txt.
1 Frame NumberFrame TypeFrame TransmissionTypeSourceNode-IDDestination
2 NARREQBroadcastNode-1BroadCastNode-118.0020.0054.00
3 NARREPPoint-to-PointNode-2Node-1Node-212.0020.8354.00
4 NAACKPoint-to-PointNANANode-114.0021.9524.0021.95
5 1DATAPoint-to-PointNode-1Node-2Node-11472.0020.0086.0
Cancel
< Back
Next >
Finish

Step 3: Click next, the following window will appear, in that select the check box as mentioned in the following window, and then click Finish button.


Text Import Wizard - Step 2 of 3
This screen lets you set the delimiters your data contains. You can see how your text is affected in the preview below.
Delimiters
Tab
Semicolon
Comma
Space
Other:
Treat consecutive delimiters as one
Text qualifier:
Data preview
Frame Number Frame Type Frame TransmissionType SourceNode-ID Destination No
NA RREQ Broadcast Node-1 BroadCast
NA RREP Point-to-Point Node-2 Node-1
NA ACK Point-to-Point NA NA
1 DATA Point-to-Point Node-1 Node-2
Cancel
< Back
Next >
Finish



	A	B	C	D	E	F	G	H
1	PACKET_ID	PACKET_TYPE	CONTROL_PACKET_TYPE	SOURCE_ID	DESTINATION_ID	TRANSMITTER_ID	RECEIVER_ID	APP_LAYER_ARRIVAL_TIME
2	0	Video	N/A	NODE-8	NODE-2	NODE-8	SWITCH-3	33333.333
3	0	Video	N/A	NODE-2	NODE-8	NODE-2	SWITCH-3	33333.333
4	0	Video	N/A	NODE-2	NODE-8	SWITCH-3	NODE-8	33333.333
5	0	Video	N/A	NODE-8	NODE-2	SWITCH-3	NODE-2	33333.333
6	1	Voice	N/A	NODE-8	NODE-2	NODE-8	SWITCH-3	40000
7	2	Voice	N/A	NODE-8	NODE-2	NODE-8	SWITCH-3	40000
8	1	Voice	N/A	NODE-8	NODE-2	SWITCH-3	NODE-2	40000
9	2	Voice	N/A	NODE-8	NODE-2	SWITCH-3	NODE-2	40000
10	0	Video	N/A	NODE-8	NODE-2	NODE-8	SWITCH-3	66666.667
11	0	Video	N/A	NODE-2	NODE-8	NODE-2	SWITCH-3	66666.667

Packet Trace Metrics:

PACKET_ID	Specifies the ID of the Data Packet
SEGMENT_ID	Specifies the ID of the segment of the Data Packet
PACKET_TYPE	Specifies the type of Packet. It can be Control Packet, Custom, Peer_to_peer, E-Mail, DataBase, FTP, Video, Voice, HTTP
CONTROL_PACKET_TYPE	Specifies the type of Control Packet transmitted. It can be TCP_SYN, TCP_SYN_ACK, TCP_ACK, WLAN_ACK, ICMP_ECHO_REQUEST, ICMP_ECHO_REPLY etc.
SOURCE_ID	Specifies the Node ID of the source device
DESTINATION_ID	Specifies the Node ID of the destination device
TRANSMITTER_ID	Specifies the Node ID of the device transmitting the packet
RECEIVER_ID	Specifies the Node ID of the device receiving the packet
APP_LAYER_ARRIVAL_TIME	Specifies the time(microseconds) at which the data packet is generated at Application Layer of source device

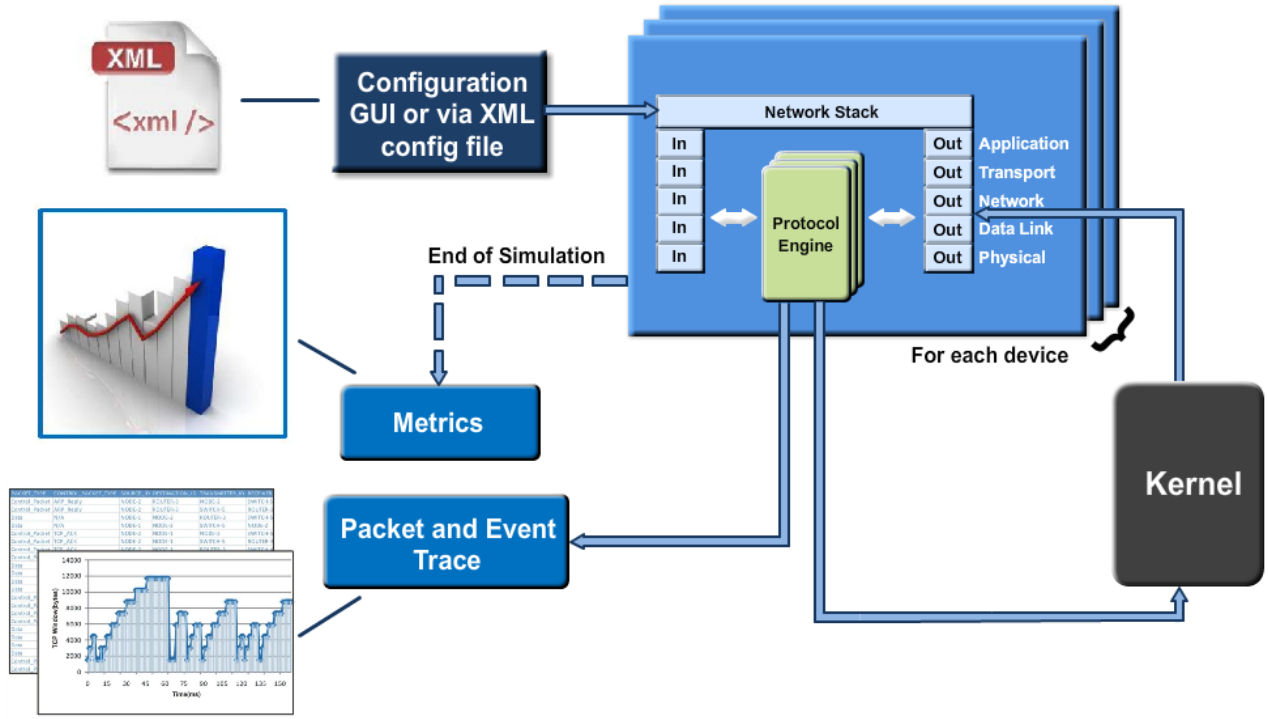
TRX_LAYER_ARRIVAL_TIME	Specifies the time(microseconds) at which the packet reaches the Transport Layer of source device
NW_LAYER_ARRIVAL_TIME	Specifies the time(microseconds) at which the packet reaches the Network Layer of source device
MAC_LAYER_ARRIVAL_TIME	Specifies the time(microseconds) at which the packet reaches the Data Link Layer of source device
PHY_LAYER_ARRIVAL_TIME	Specifies the time(microseconds) at which the packet reaches the Physical Layer of source device
PHY_LAYER_START_TIME	Specifies the time(microseconds) at which the packet leaves the Physical Layer of source device
PHY_LAYER_END_TIME	Specifies the time(microseconds) at which the packet reaches the Physical Layer of destination device
APP_LAYER_PAYLOAD	Specifies the size of the Payload at Application Layer
TRX_LAYER_PAYLOAD	Specifies the size of the Payload at Transport Layer
NW_LAYER_PAYLOAD	Specifies the size of the Payload at Network Layer
MAC_LAYER_PAYLOAD	Specifies the size of the Payload at Data Link Layer
PHY_LAYER_PAYLOAD	Specifies the size of the Payload at Physical Layer
PHY_LAYER_OVERHEAD	Specifies the size of the overhead in Physical layer
PACKET_STATUS	Specifies whether the Packet is Successful, Collided or Errored
LOCAL_ADDRESS	Specifies the Port Number at Source Node
FOREIGN_ADDRESS	Specifies the Port Number at Destination Node
CWND	Specifies the size of congestion window
SEQ_NO	If TCP is enabled, it specifies the TCP Sequence number of the packet
ACK_NO	If TCP is enabled, it specifies the TCP Acknowledgement number of the packet
RTT	Specifies the Round Trip Time for the packet
RTO	Specifies the Retransmission Timeouts
CONNECTION_STATE	Specifies the state of TCP connection
CW	Specifies the maximum size of contention window
BO_TIME	Specifies the Backoff time (in microseconds)
DATARATE	In case of WLAN, it specifies the data rate at which the packet was transmitted
RX_POWER	Specifies the power(in dBm) required by the device at reception
RX_SENSITIVITY	Specifies the minimum magnitude (in dBm) of input signal required to produce a specified output signal.

Generating Event Trace (Available in Standard and Pro Version only)

NetSim Network Stack:

NetSim's Network Stack forms the core of NetSim and its architectural aspects are diagrammatically explained below. Network Stack accepts inputs from the end-user in the form of Configuration file and the data flows as packets from one layer to another layer in the Network Stack.

All packets, when transferred between devices move up and down the stack, and all events in NetSim fall under one of these ten categories of events, namely, ***Physical IN, Data Link IN, Network IN, Transport IN, Application IN, Application Out, Transport OUT, Network OUT, Data Link OUT*** and ***Physical OUT***. The IN events occur when the packets are entering a device while the OUT events occur while the packet is leaving a device.



Every device in NetSim has an instance of the Network Stack shown above. Switches & Access points have a 2 layer stack, while routers have a 3 layer stack. End-nodes have a 5 layer stack.

The protocol engines are called based on the layer at which the protocols operate. For example, TCP is called during execution of Transport IN or Transport OUT events, while 802.11b WLAN is called during execution of MAC IN, MAC OUT, PHY IN and PHY OUT events.

When these protocols are in operation they in turn generate events for NetSim's discrete event engine to process. These are known as SUB EVENTS. All SUB EVENTS, fall into one of the above 10 types of EVENTS.

Each event gets added in the Simulation kernel by the protocol operating at the particular layer of the Network Stack. The required sub events are passed into the Simulation kernel. These sub events are then fetched by the Network Stack in order to execute the functionality of each protocol. At the end of Simulation, Network Stack writes trace files and the Metrics files that assist the user in analyzing the performance metrics and statistical analysis.

Event Trace:

The event trace records every single event along with associated information such as time stamp, event ID, event type etc in a text file or .csv file which can be stored at a user defined location.

Apart from a host of information, the event trace has two special information fields for diagnostics

a) A log of the file name and line number from where the event was generated (Please refer Help "**Event Trace -Writing file name and line number of the code which generated the event**") and

b) Previous event which triggered the current event.

Note: *Turning on Event Trace will slow down the simulation significantly*

How to enable Event Trace via GUI?

If NetSim runs via GUI, event trace can be turned on by clicking the Event Trace icon in the tool bar in the Internetworks menu and selecting the required fields in the event trace.

How to enable Event Trace via CLI?

If NetSim runs via CLI, then the event trace can be turned on by enabling the event trace in the STATISTICS_COLLECTION tag of the configuration file.

How to import Event Trace to Excel?

Refer Help on "**Generating Packet Trace How to import Packet Trace to Excel?"**

Event Trace Metrics:

Event_Id	Specifies the ID of the Event
Event_Type	Specifies the type of event being performed, for eg - APPLICATION_IN, APPLICATION_OUT, MAC_OUT, MAC_IN, PHYSICAL_OUT, PHYSICAL_IN,etc
Event_Time	Specifies the time(in microseconds) at which the event is being executed
Device_Type	Specifies the type of device in which the current event is being executed
Device_Id	Specifies the ID of device in which the current event is being executed
Interface_Id	Specifies the Interface_Id of device in which the present event is being executed.
Application_Id	Specifies the ID of the Application on which the specific event is executed
Packet_Id	Specifies the ID of the packet on which the current event is being executed
Segment_Id	Specifies the ID of the segment of packet on which the current event is being executed
Protocol_Name	Specifies the Protocol which is presently executed
Subevent_Type	Specifies the protocol sub event which is being executed. If the sub event value is 0, it indicates interlayer communication (Ex: MAC_OUT called by NETWORK_OUT) or a TIMER_EVENT which has no sub event.
Packet_Size	Specifies the size of packet during the current event
Prev_Event_Id	Specifies the ID of the event which generated the current event.

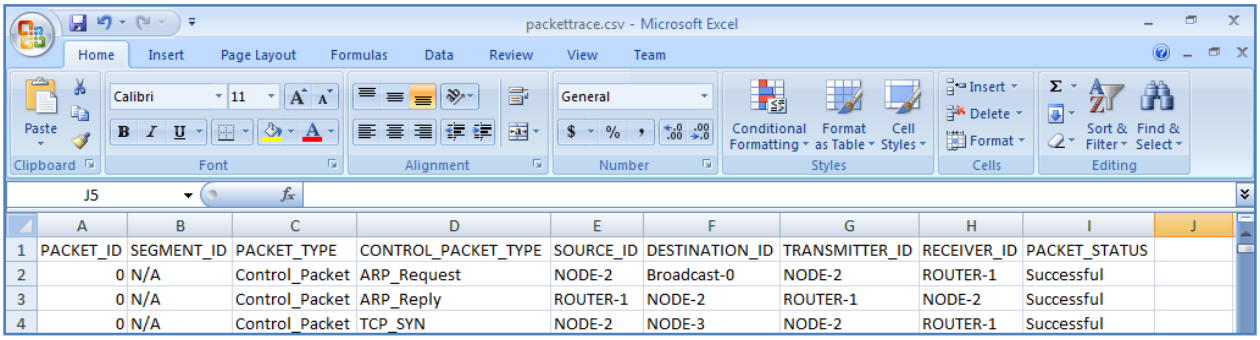
Trace data analysis techniques

This section explains about how to analyze the data of a NetSim trace file.

How to set filters to NetSim trace file

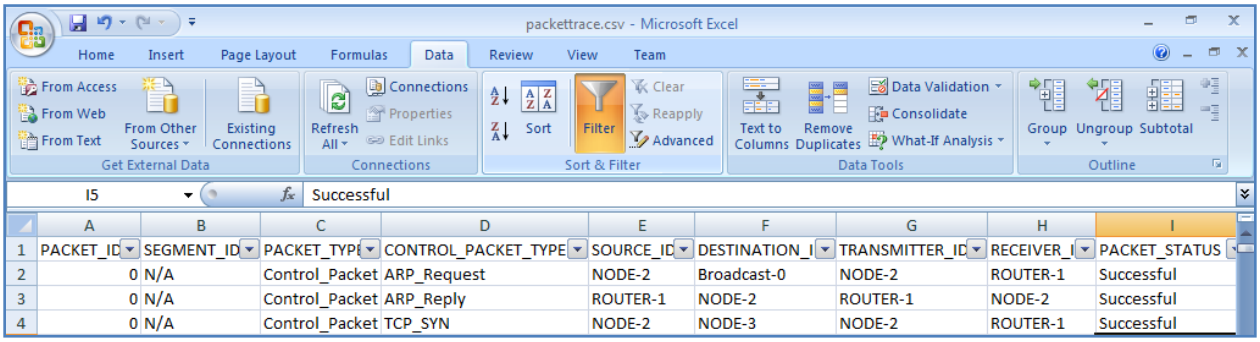
Step 1: Open the trace file. (In this example packet trace is opened)

> ***Note:*** *Refer "NetSim Help->Generating Packet trace" and "NetSim Help->Generating Event trace" Sections to generate the trace files.*



	A	B	C	D	E	F	G	H	I	J
1	PACKET_ID	SEGMENT_ID	PACKET_TYPE	CONTROL_PACKET_TYPE	SOURCE_ID	DESTINATION_ID	TRANSMITTER_ID	RECEIVER_ID	PACKET_STATUS	
2	0	N/A	Control_Packet	ARP_Request	NODE-2	Broadcast-0	NODE-2	ROUTER-1	Successful	
3	0	N/A	Control_Packet	ARP_Reply	ROUTER-1	NODE-2	ROUTER-1	NODE-2	Successful	
4	0	N/A	Control_Packet	TCP_SYN	NODE-2	NODE-3	NODE-2	ROUTER-1	Successful	

Step 2: Go to Data menu and click on filter icon to enable the auto filter.

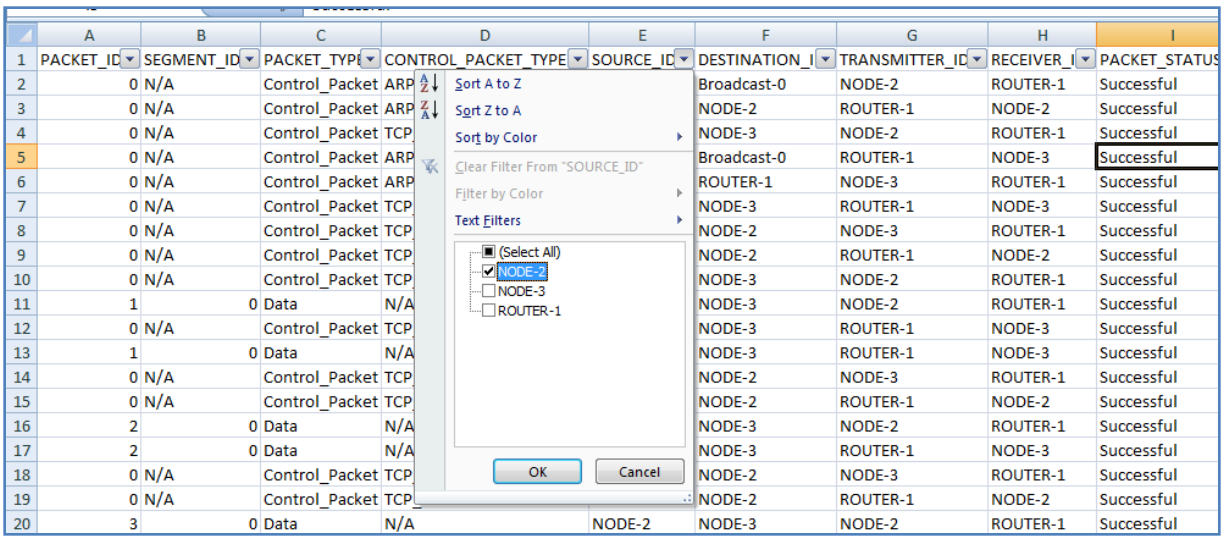


	A	B	C	D	E	F	G	H	I
1	PACKET_ID	SEGMENT_ID	PACKET_TYPE	CONTROL_PACKET_TYPE	SOURCE_ID	DESTINATION_I	TRANSMITTER_ID	RECEIVER_I	PACKET_STATUS
2	0	N/A	Control_Packet	ARP_Request	NODE-2	Broadcast-0	NODE-2	ROUTER-1	Successful
3	0	N/A	Control_Packet	ARP_Reply	ROUTER-1	NODE-2	ROUTER-1	NODE-2	Successful
4	0	N/A	Control_Packet	TCP_SYN	NODE-2	NODE-3	NODE-2	ROUTER-1	Successful

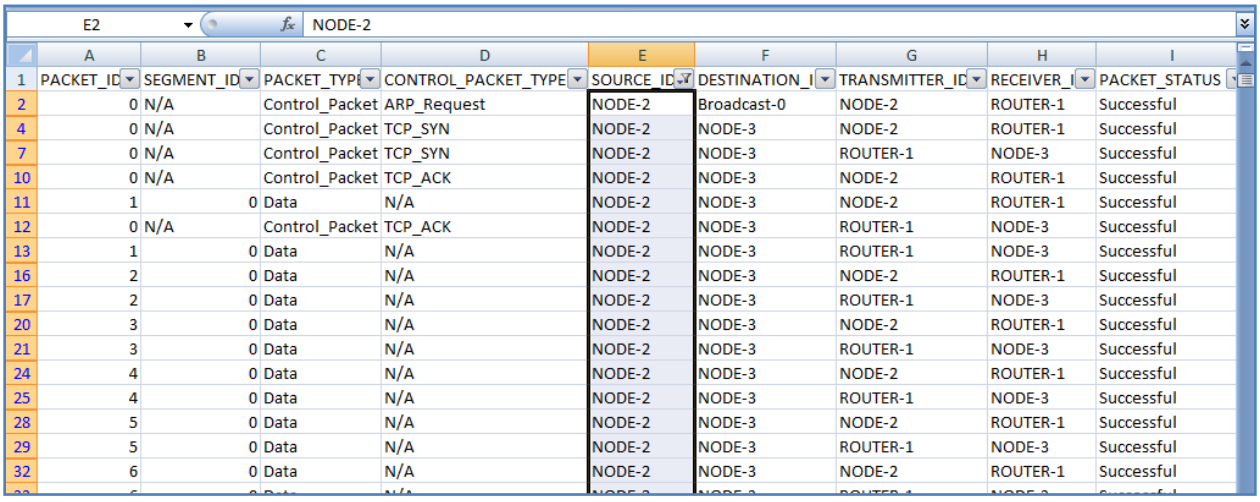
Step 3: Click the arrow in the header of the column you want to filter. In the list of text or numbers, uncheck the (Select All) box at the top of the list, and then check the boxes of the items you want to show.

For example, click on arrow of SOURCE_ID and uncheck the "Select all" check box and select NODE 2 then click on ok.

All the rows which are having NODE 2 as source id will be shown.



	A	B	C	D	E	F	G	H	I
1	PACKET_ID	SEGMENT_ID	PACKET_TYPE	CONTROL_PACKET_TYPE	SOURCE_ID	DESTINATION_I	TRANSMITTER_ID	RECEIVER_I	PACKET_STATUS
2	0	N/A	Control_Packet	ARP		Broadcast-0	NODE-2	ROUTER-1	Successful
3	0	N/A	Control_Packet	ARP		NODE-2	ROUTER-1	NODE-2	Successful
4	0	N/A	Control_Packet	TCP		NODE-3	NODE-2	ROUTER-1	Successful
5	0	N/A	Control_Packet	ARP		Broadcast-0	ROUTER-1	NODE-3	Successful
6	0	N/A	Control_Packet	ARP		ROUTER-1	NODE-3	ROUTER-1	Successful
7	0	N/A	Control_Packet	TCP		NODE-3	ROUTER-1	NODE-3	Successful
8	0	N/A	Control_Packet	TCP		NODE-2	NODE-3	ROUTER-1	Successful
9	0	N/A	Control_Packet	TCP		NODE-2	ROUTER-1	NODE-2	Successful
10	0	N/A	Control_Packet	TCP		NODE-3	NODE-2	ROUTER-1	Successful
11	1	0	Data	N/A		NODE-3	NODE-2	ROUTER-1	Successful
12	0	N/A	Control_Packet	TCP		NODE-3	ROUTER-1	NODE-3	Successful
13	1	0	Data	N/A		NODE-3	ROUTER-1	NODE-3	Successful
14	0	N/A	Control_Packet	TCP		NODE-2	NODE-3	ROUTER-1	Successful
15	0	N/A	Control_Packet	TCP		NODE-2	ROUTER-1	NODE-2	Successful
16	2	0	Data	N/A		NODE-3	NODE-2	ROUTER-1	Successful
17	2	0	Data	N/A		NODE-3	ROUTER-1	NODE-3	Successful
18	0	N/A	Control_Packet	TCP		NODE-2	NODE-3	ROUTER-1	Successful
19	0	N/A	Control_Packet	TCP		NODE-2	ROUTER-1	NODE-2	Successful
20	3	0	Data	N/A	NODE-2	NODE-3	NODE-2	ROUTER-1	Successful



E2 NODE-2

	A	B	C	D	E	F	G	H	I
1	PACKET_ID	SEGMENT_ID	PACKET_TYPE	CONTROL_PACKET_TYPE	SOURCE_ID	DESTINATION_I	TRANSMITTER_ID	RECEIVER_I	PACKET_STATUS
2	0	N/A	Control_Packet	ARP_Request	NODE-2	Broadcast-0	NODE-2	ROUTER-1	Successful
4	0	N/A	Control_Packet	TCP_SYN	NODE-2	NODE-3	NODE-2	ROUTER-1	Successful
7	0	N/A	Control_Packet	TCP_SYN	NODE-2	NODE-3	ROUTER-1	NODE-3	Successful
10	0	N/A	Control_Packet	TCP_ACK	NODE-2	NODE-3	NODE-2	ROUTER-1	Successful
11	1	0	Data	N/A	NODE-2	NODE-3	NODE-2	ROUTER-1	Successful
12	0	N/A	Control_Packet	TCP_ACK	NODE-2	NODE-3	ROUTER-1	NODE-3	Successful
13	1	0	Data	N/A	NODE-2	NODE-3	ROUTER-1	NODE-3	Successful
16	2	0	Data	N/A	NODE-2	NODE-3	NODE-2	ROUTER-1	Successful
17	2	0	Data	N/A	NODE-2	NODE-3	ROUTER-1	NODE-3	Successful
20	3	0	Data	N/A	NODE-2	NODE-3	NODE-2	ROUTER-1	Successful
21	3	0	Data	N/A	NODE-2	NODE-3	ROUTER-1	NODE-3	Successful
24	4	0	Data	N/A	NODE-2	NODE-3	NODE-2	ROUTER-1	Successful
25	4	0	Data	N/A	NODE-2	NODE-3	ROUTER-1	NODE-3	Successful
28	5	0	Data	N/A	NODE-2	NODE-3	NODE-2	ROUTER-1	Successful
29	5	0	Data	N/A	NODE-2	NODE-3	ROUTER-1	NODE-3	Successful
32	6	0	Data	N/A	NODE-2	NODE-3	NODE-2	ROUTER-1	Successful

Typically filters can be set to observe “Errored/Collided/Successful “packets, packets of destination and packets of source.

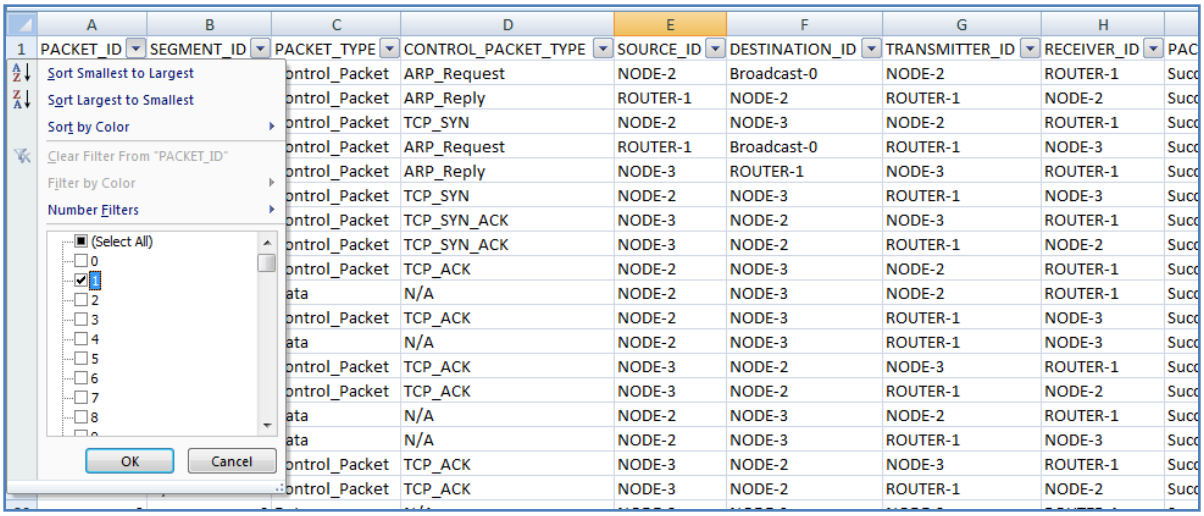
Observing packet flow in the Network through packet trace file

Step 1: Open the packet trace file.

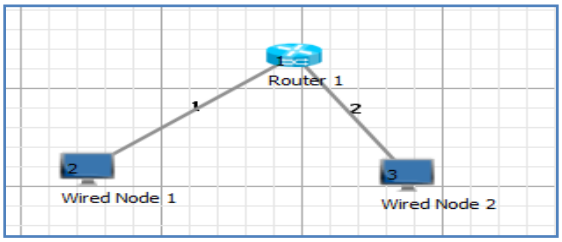
Step 2: Set the filter.

> ***Note:*** *Refer “NetSim Help->Trace data Analysis techniques->How to set filter to a NetSim trace file” section to set the filter.*

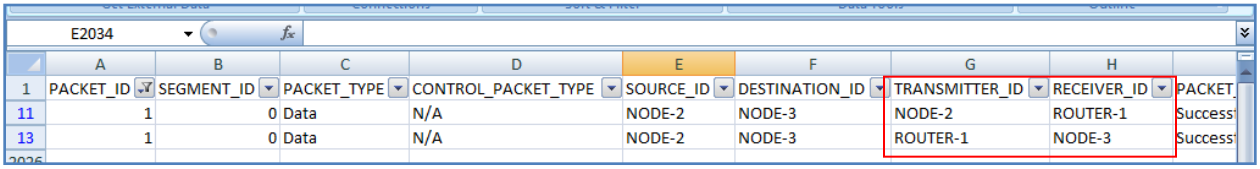
Step 3: Click the arrow in the header of the column PACKET_ID and uncheck the “Select all” check box and select the packet id which you want to observe, for example 1, and then click on ok.



Scenario is as shown below and traffic flow is from Wired Node 1 to Wired Node 2.



Flow of packet 1 can be observed from the packet trace as shown below.



PACKET_ID	SEGMENT_ID	PACKET_TYPE	CONTROL_PACKET_TYPE	SOURCE_ID	DESTINATION_ID	TRANSMITTER_ID	RECEIVER_ID	PACKET
1	0	Data	N/A	NODE-2	NODE-3	NODE-2	ROUTER-1	Success
1	0	Data	N/A	NODE-2	NODE-3	ROUTER-1	NODE-3	Success

Note: *In the trace file device IDs are shown not device names. Wired Node 1's ID is 2 so it is Shown as NODE-2, Wired Node 2's ID is 3 so it is shown as NODE -3, Router-1' ID is 1 so it is shown as ROUTER-1. Device IDs are shown on the top of the device icon in the above scenario.*

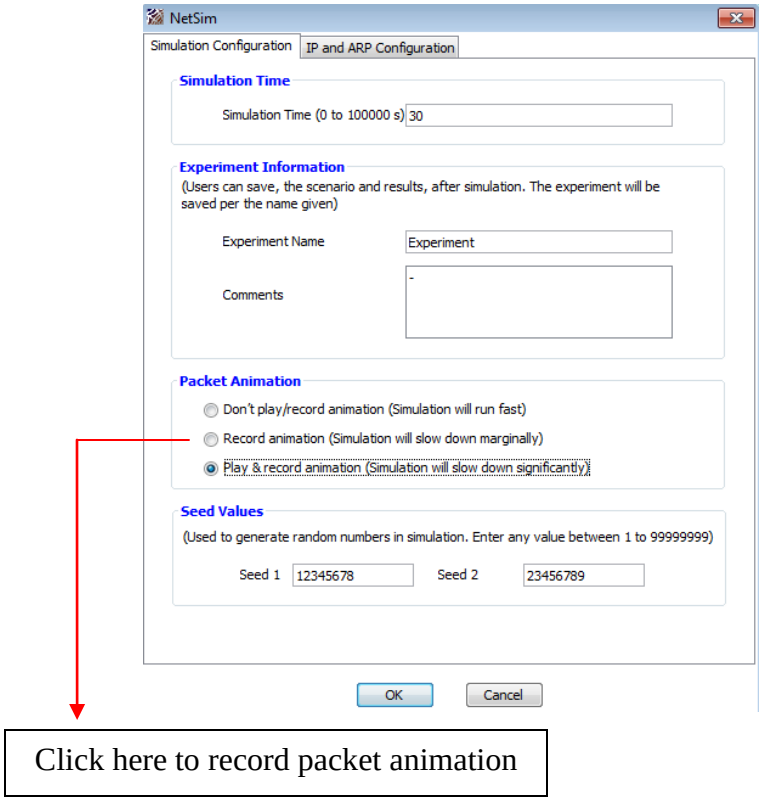
In a scenario source and destinations are fixed but transmitter and receiver are changed. For example in the above scenario NODE-2 is the source and NODE-3 is the destination, but when NODE- 2 sending the packet to the ROUTER-1 then NODE-2 is the transmitter and ROUTER-1 is the receiver. When ROUTER-1 sending the packet to the NODE-3, ROUTER-1 is the transmitter and NODE-3 is the receiver.

Packet Animation

NetSim provides the feature to play and record animations to the user. Packet animation enables users to watch traffic flow through the network for in-depth visualization and analysis.

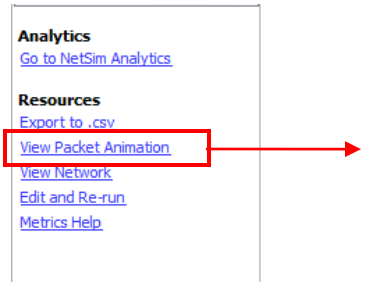
User has the following options before running simulation:

- No animation,
- Record the animation and
- Play and record animation while running simulation.


NetSim
Simulation Configuration
IP and ARP Configuration
Simulation Time
Simulation Time (0 to 100000 s) 30
Experiment Information
(Users can save, the scenario and results, after simulation. The experiment will be saved per the name given)
Experiment Name
Experiment
Comments
Packet Animation
Don't play/record animation (Simulation will run fast)
Record animation (Simulation will slow down marginally)
Play & record animation (Simulation will slow down significantly)
Seed Values
(Used to generate random numbers in simulation. Enter any value between 1 to 99999999)
Seed 1
12345678
Seed 2
23456789
OK
Cancel

Click here to record packet animation

The packet animation would then be recorded and the user can view the animation from the metrics screen as shown below:

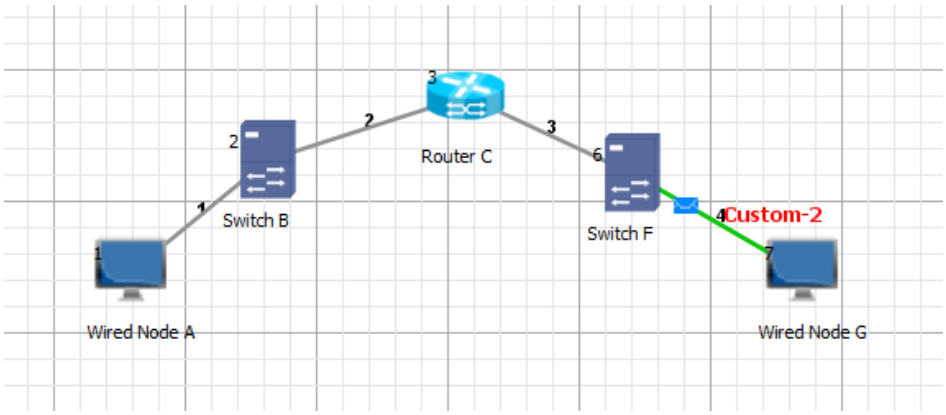


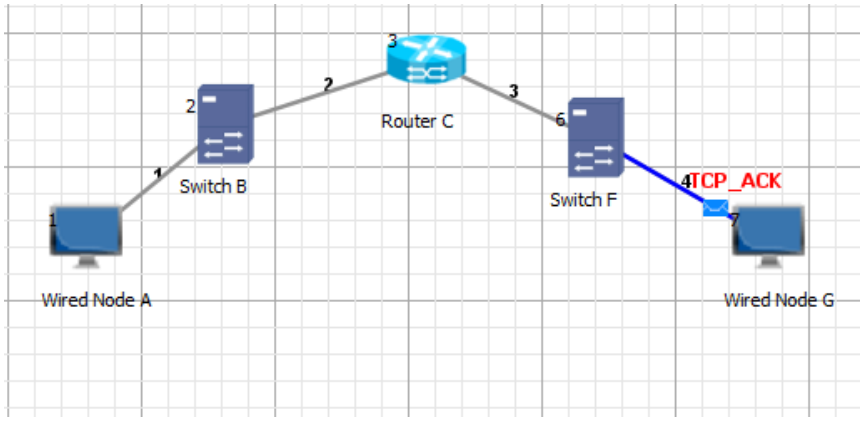
Click here to view packet animation

When packet animation is enabled, a screen with the following toolbar appears:

While viewing packet animation, user can see the flow of packets as well as the type of packet. Blue color packet denotes control packet, green color is used for data packet and red color is error/collided packet.

Example showing packet animation: In first figure, Custom data packet is flowing from Switch F to Node G (green color) and TCP_ACK is sent from Node G to Switch F in second figure (blue color).



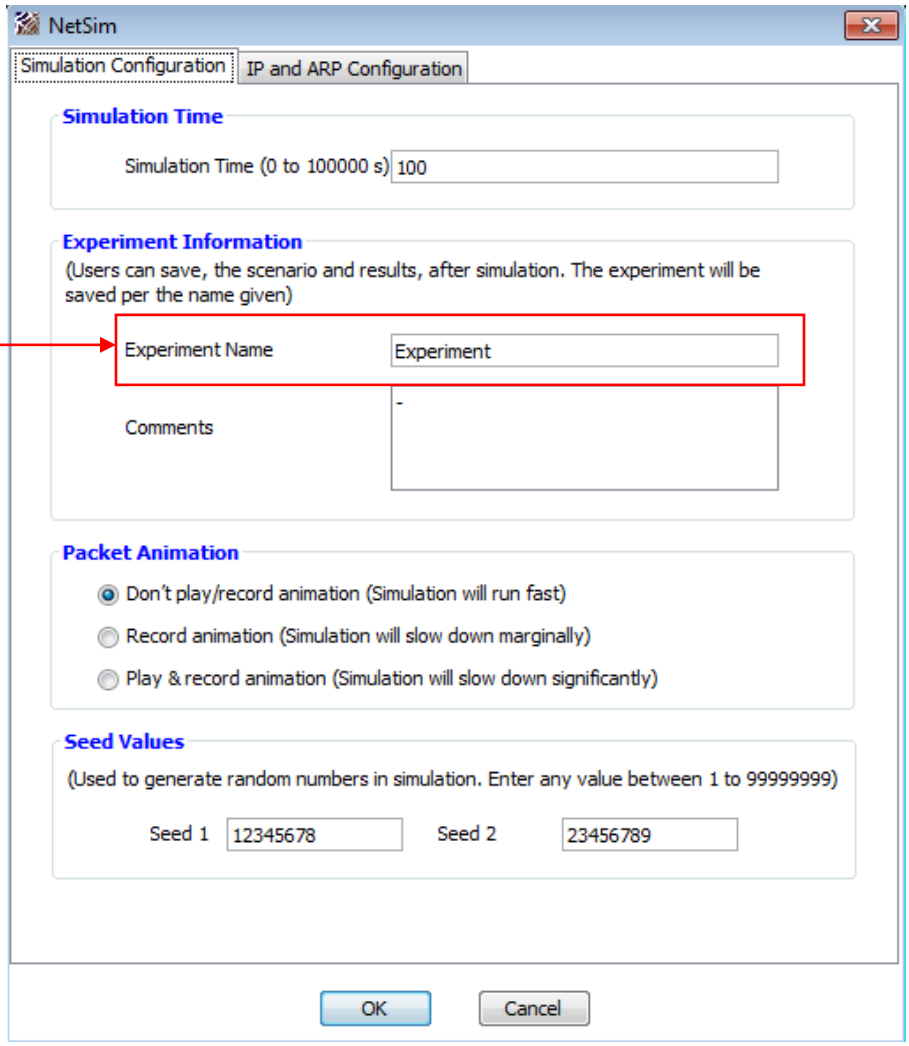


Saving Experiments

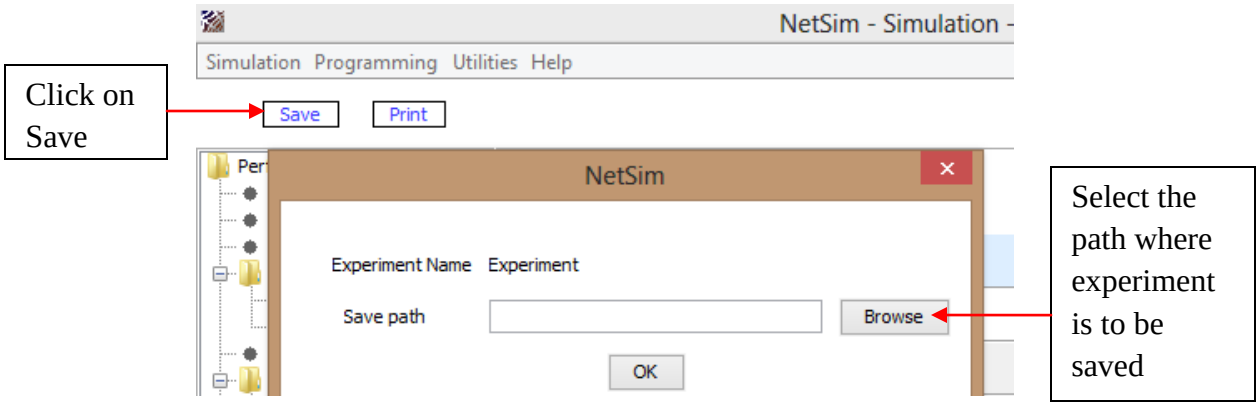
For Internetworks, BGP Networks, Advanced Wireless Networks–MANET, Wireless Sensor Networks, Personal Area Networks, Cognitive Radio Networks and LTE networks:

Step 1: Just before simulating the network, enter the Experiment name with which you want to save the experiment and any additional comments (optional).

Experiment will be saved in this name. The file location can be set after simulation



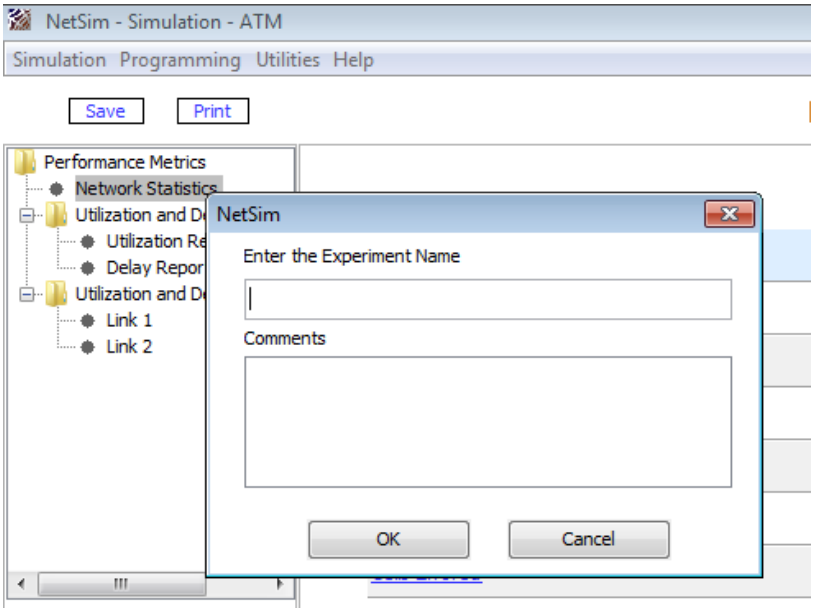
Step 2: Click on save button after completing the simulation, browse the path where the experiment needs to be saved and click OK. The configuration file and metrics file associated to this experiment is saved in the location provided.



Note:

1. *Experiment Name can be changed before running the simulation only.*
2. *Saving the Experiment in same location without changing name will lead to overwriting the saved experiment.*

For Legacy / MPLS / Wi-Max Networks, the following screen appears and the **Experiment**is saved by entering the experiment name and clicking OK:



Opening Saved Experiments

Open Network

Select **Network** from **Simulation →Open** menu to open saved experiments. The following steps need to be followed:

- **Select** the **User** (*Note*: This option is available in Admin login only)
- **Select** the **Network**. There are various options like Internetworks, Legacy Networks, Advanced Wireless Networks, MPLS Networks, BGP Networks, Cellular Networks, Wireless Sensor Networks, Personal Area Networks, LTE Networks and Cognitive Radio Networks will be available. Choose based on the saved experiment that needs to be opened.
- **Select** the **Protocol**(*Note:* This is not applicable for Internetworks, BGP Networks, Wireless Sensor Networks, Personal Area Networks, LTE Networks and Cognitive Radio Networks)
- **Select** the **Config File/ Experiment**
  - For Internetworks, BGP Networks, Advanced Wireless Networks – MANET, Wireless Sensor Networks, Personal Area Networks, LTE networks and Cognitive Radio Networks and **Select** the **Config File** of the particular **Experiment** that needs to be opened using **Browse** button.
  - For Other Networks, **select** the **Experiment** that needs to be opened.
- **Click** on **Ok** button to **open** the specified **Experiment**. **User** can modify the existing **Scenario, Simulate** and **Save** it. Else **Click** on **Cancel** button to **Exit** the screen.

Open Metrics

Select **Metrics** from **Simulation →Open** menu to open metrics results of the saved experiments. The following steps need to be followed,

- **Select** the **User** (*Note*: This option is available in Admin login only. This option can be used to open experiments created by different users)
- **Select** the **Network**: There are several options like Internetworks, Legacy Networks, Advanced Wireless Networks, MPLS Networks, BGP Networks, Cellular Networks,

Wireless Sensor Networks, Personal Area Networks, LTE Networks and Cognitive Radio Networks available, choose based on the saved experiment that needs to be opened.

- **Select** the **Protocol** (*Note:* This is not applicable for Internetworks, BGP Networks, Wireless Sensor Networks, Personal Area Networks, LTE Networks and Cognitive Radio Networks)
- **Select** the **Metrics File**/ **Experiment**
  - For Internetworks, BGP Networks, Advanced Wireless Networks, Wireless Sensor Networks, Personal Area Networks, LTE Networks and Cognitive Radio Networks and **Select** the **Metrics File** of the particular **Experiment** that needs to be opened using **Browse** button.
  - For Other Networks, **select** the **Experiment** that needs to be opened.
- **Click** on **Ok** button to **open** the selected **Experiment**. **User** can modify the existing **Scenario, Simulate** and **Save** it. Else **Click** on **Cancel** button to **Exit** the screen.

Deleting Saved Experiments

Delete

In the **Simulation** menu select **Delete** option to delete the saved **Simulated Experiments**. To delete **Experiments** the user needs to follow the below given steps,

- **Click** the **Delete/Delete All** button to **enter** into the next window.
  - **Delete – Click** the **Delete** button to delete single experiment.
  - **Delete All – Click** the **Delete All** button to delete all the saved experiments.
- **In the next window,**
  - **Delete – Select** the **Experiment Name** that needs to be deleted.
  - **Delete All – Select** the **User Name** in order to delete the experiments under that user.(*Note:* This option will be available only in Admin login)
  - **Click** on **Delete** button to delete the **Experiment(s)**, else **click** on **Cancel** button to **cancel** deletion.
  - After clicking on the **Delete** button a message appears which says "**Are you sure you want to delete this Experiment(s)?**" **Click** on **Yes** to continue deleting the **Experiment(s)**, else **click** on **No** to **cancel** deleting.
  - **Click** on **OK** button in the "**The experiment is deleted**" or "**The experiments have been deleted"** window.

Note:

- **Admin User:** An option called **User Name** is available only in when the User has logged into Admin. This option can be made use for deleting experiments under any User.
- **Local User:** A User who has a normal privilege other than Admin can delete experiments only which are created by that particular user.
- In the case of Internetworks, Advanced Wireless Networks – MANET, BGP Networks, Wireless Sensor Networks, Personal Area Networks, Cellular Networks, LTE Networks and Cognitive Radio Networks the experiments can be saved by the user at any desired local on the computer. These experiments cannot be deleted via NetSim, and must be directly deleted from the computer.

Analytics

Under the **Simulation** menu click **Analytics** to view the **Analytics** screen. This module is designed to enable comparison for different performance metrics. A graphing engineer facilitates plotting of different performance metrics across the different experiments. This can be used to infer/trend how changes in input parameters affect network performance. An explanation of the use of the different fields in the **Analytics** screen is detailed below:

- **User:** This Option can be used to compare different experiments created by a single user. As Admin is a Super User, he/she can compare all the experiments created by all the users. Things that cannot be done are,
    - User to User comparison cannot be made, i.e. two or more different experiments created by two or more different users cannot be compared.
    - Experiments of a particular Network cannot be compared with experiments of other Networks.

- In **Tool Bar**, there are various tabs available like Internetworks, Legacy Networks, BGP Networks, MPLS Networks, Advanced Wireless Networks - MANET, Advanced Wireless Networks – WiMAX, Cellular Networks, Wireless Sensor Networks, Personal Area Networks, Cognitive Radio Networks and LTE networks. Click on the particular **Network tab** for comparing the performance of protocols under that Network.
    - For Internetworks, Advanced Wireless Networks – MANET, BGP Networks,Wireless Sensor Networks, Personal Area Networks, LTE networks and Cognitive Radio Networks, **select the Metrics File**–**Select** the **Metrics File**to add it onto the Metrics Table by using **Browse** button.
    - For Other Networks, **select the Experiment** based on the tab selected. When one of the tabs is selected, all the experiments saved under the particular **Network** will be listed. Click on the Experiment Name to add it onto the Metrics Table.
    - **Export to .csv - Click** on **Export to .csv** to export the **Metrics Table** to a "**.csv**" format. This action button will export the contents to a ".csv" format that is available.

- **Print** - **Click** on the **Print** option to print the **Metrics Table**. This action button will fetch the Print dialogue box.

- **The Graph Field**
  - **Select the Metrics** - **Select** the coordinates for **Y-axis** by clicking on the dropdown menu.
  - **Graph** - Based on the **X-axis** (i.e. **Metrics File**/ **Experiment** selected) and **Y-axis** (i.e. **Metrics** selected by using the dropdown menu above the **graph**), a **Bar graph** will be plotted.
  - Add metrics file of different experiments to compare.

- **The Metrics Table-** This is the table that is generated when the Experiments are selected using the **Select the Metrics File**/ **Experiment** option.

  - The first column in the **Metrics Table** consists of the **Check Boxes** that are used to **select** (i.e. by default all the Experiments are selected) or **deselect** the selected **Experiments**. If an Experiment is selected, then that Experiment would be compared along with other Experiments, else the Experiment would not be compared with other Experiments.

  - The second column in the **Metrics Table** "**X**" button is available. This "**X**" button is used for deleting the **Experiment.**

Running Simulation via Command Line Interface (CLI)

Note:*This feature is available from v7 onwards. In v8, NetSim can be run via CLI for Internetworks, Advanced Wireless Networks – MANET, Wireless Sensor Networks, Personal Area Networks, BGP Networks,Cellular Networks, LTE Networks and Cognitive Radio Networks only. For other networks NetSim v8 can only be run via GUI.*

Advanced users can model their simulation via a config file (which can be created without the NetSim GUI) and run the simulation from command line. This is typically done in cases where very large networks are to be simulated (it takes too long to create it in the GUI), or to run a series of simulations automatically (Ref Help on NetSim Batch Processing). The config file contains all required information to run the simulation including the network topology, devices, links, traffic, statistics, traces etc.

To run Simulation in NetSim through command line interface (CLI), the following steps have to be followed.

Step 1: Note the App Path

App path is the file location in the system in which NetSim has been installed.

The app path can be found out by right clicking the NetSim Shortcut in the desktop and selecting Open file location (In Vista and Windows 7) or Find Target (In Windows XP). The app path will be something like "C:\Program Files\NetSim Standard\bin", depending on where NetSim is installed.

Note: The path to the bin folder of the app path must be mentioned

Step 2: Note the IO Path

IO path (Input/output Path) is the path where the input and output files of an application is written. This is similar to the temp path of windows OS. For NetSim, the IO path can be got by Start -> Run-> %temp%/NetSim. Once you reach this folder, the user can notice that the path would be something like "C:\Users\Ram\AppData\Local\Temp\NetSim"

Note: *The IO path must have write permission. For example, Windows 7 does not provide write permission for the app path, explicitly. NetSim may exhibit unpredictable behavior if this path does not have write permission.*

The IO path is the path where the ***Configuration.xml*** (NetSim Config file)of the scenario, that will be simulated, should be present.

App path and IO path can also be same, i.e., Configuration.xml can be placed inside the app path (if the app path has write permission). Otherwise, users can create a folder for IO path and Configuration.xml can be placed inside that folder.

Note: *Sample configuration.xml files are available in the* ***<apppath>/Docs/Sample_Configurations*** *folder of the NetSim install directory. Rename the sample file as "Configuration.xml" before running NetSim through CLI. Depending on your system setting, the extension (.xml) might be implicitly present and not visible. In such cases do not add the extension, as that will cause the file to be named as configuration.xml.*

Step 3: Running NetSim through command line for Simulation

To run NetSim through command line, copy the app path where NetSimCore.exe is present and paste it in the command prompt.

```
>cd <app path>
```

For floating/roaming licenses, type the following in the command prompt

```
>NetSimCore.exe<space> -apppath<space><app path><space>-iopath<space><io path><space>-license<space>5053@<Server IP Address>
```

Where,

- <app path> contains all files of NetSim including NetSimCore.exe
- <iopath> contains Configuration.xml and Configuration.xsd

- 5053 is the port number through which the system communicates with the license server i.e the system in which the dongle is running (for floating license users)
- <Server IP Address> is the ip address of the system where NetSim license server (dongle) is running. Please contact your network administrator / lab in charge to know the IP address of the PC where the NetSim license server is running.

The following screenshot is the example of running NetSim through CLI where the ip address of the NetSim license server is 192.168.0.2

Administrator: Command Prompt
C:\ProgramFiles\NetSimStandard\bin>NetSimCore.exe -apppath C:\ProgramFiles\NetSimStandard\bin -iopath C:\Users\ADMINI~1\AppData\Local\Temp\NetSim -license 5053@192.168.0.2_

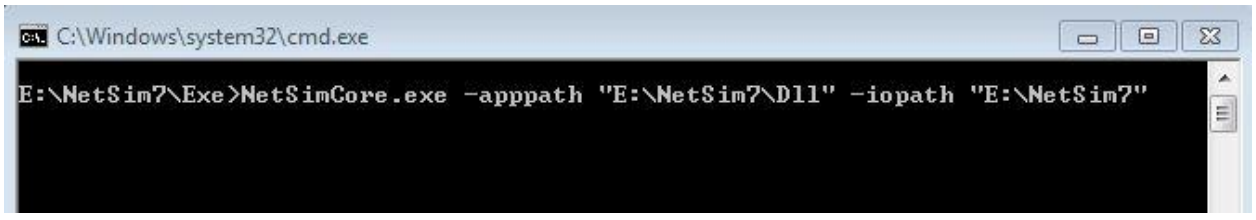
For node-locked licenses, type the following in the command prompt

>NetSimCore.exe<space> -apppath<space><app path><space>-iopath<space><io path><space>

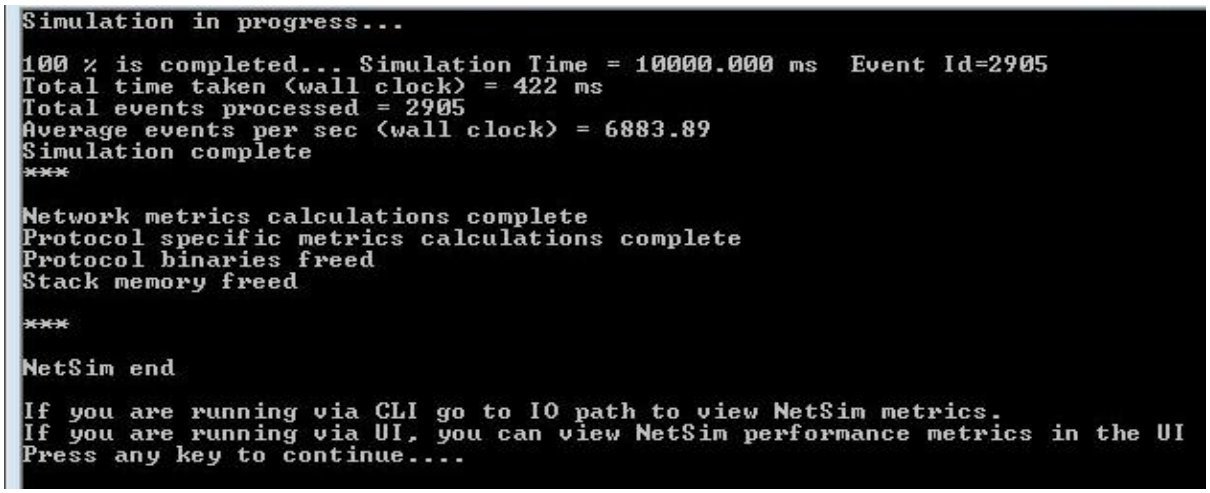
Where,

- <app path> contains all files of NetSim including NetSimCore.exe
- <iopath> contains Configuration.xml and Configuration.xsd

The following screenshot is the example of running NetSim through CLI for the node locked license.



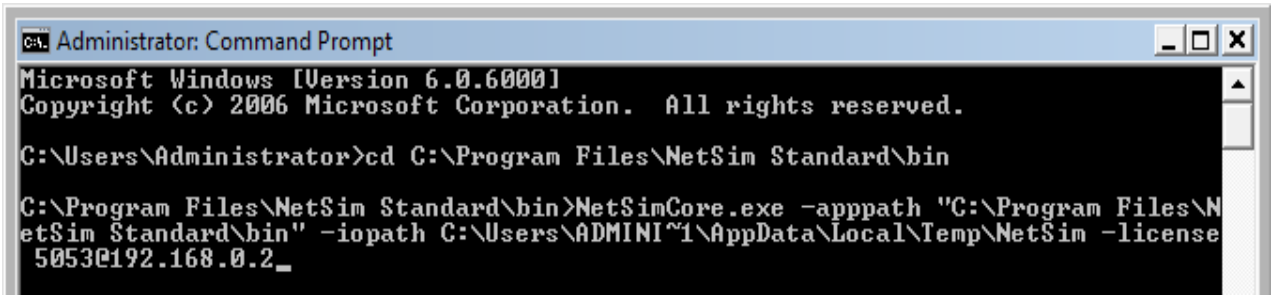
Simulation will be completed successfully and the text files that are requested by the end user in Configuration.xml will be written in the iopath.


Simulation in progress...
100 % is completed... Simulation Time = 10000.000 ms Event Id=2905
Total time taken (wall clock) = 422 ms
Total events processed = 2905
Average events per sec (wall clock) = 6883.89
Simulation complete
***
Network metrics calculations complete
Protocol specific metrics calculations complete
Protocol binaries freed
Stack memory freed
***
NetSim end
If you are running via CLI go to IO path to view NetSim metrics.
If you are running via UI, you can view NetSim performance metrics in the UI
Press any key to continue....

Note: If the folder name contains white space, then mention the folder path within double quotes while specifying the folder name in the command prompt. For example, if app path contains white space, then the app path must be mentioned within double quotes in the command prompt as given below.

>cd <app path>

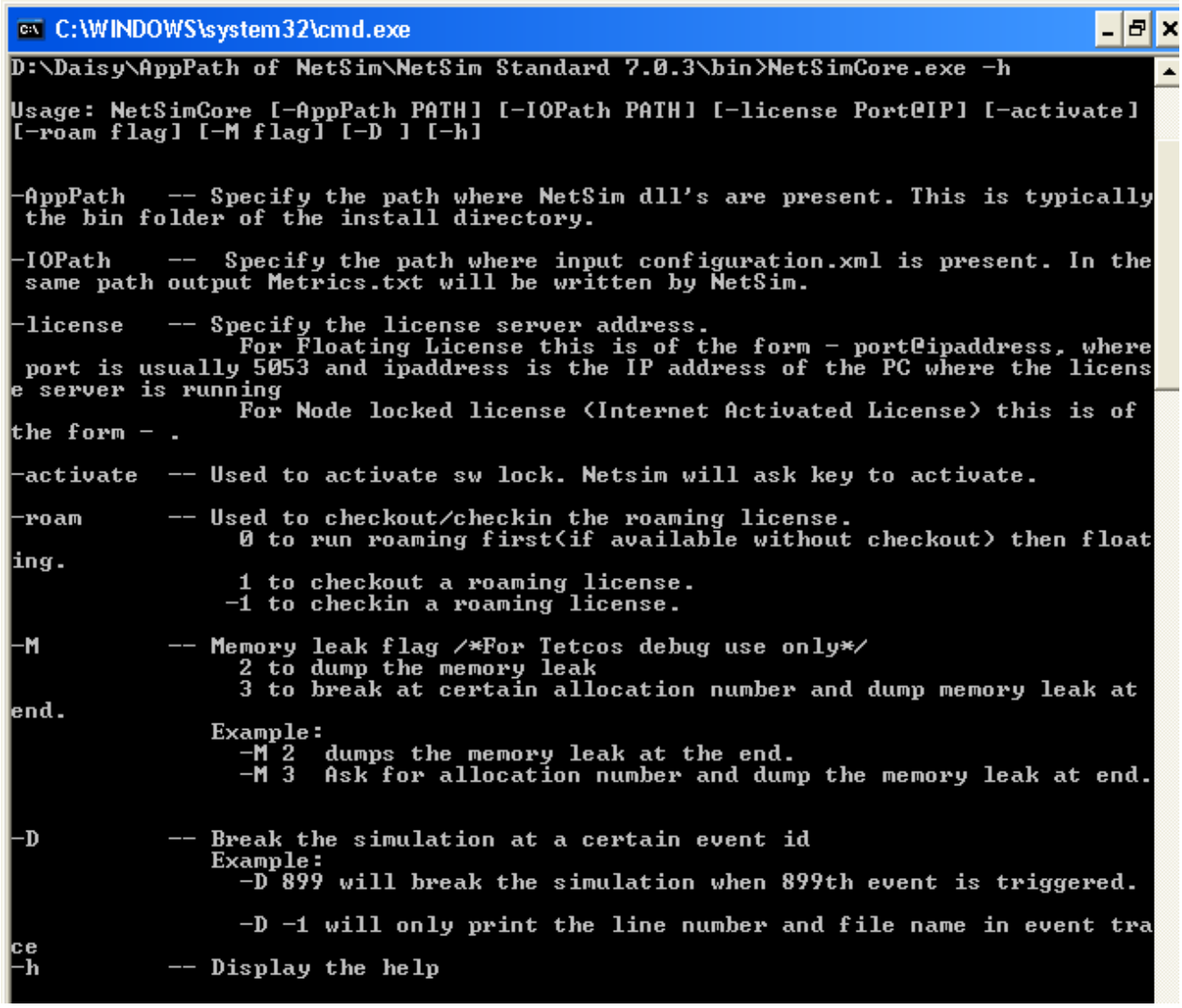
>NetSimCore.exe -apppath <"app path"> -iopath<io path>-license 5053@<Server IP Address>


Administrator: Command Prompt
Microsoft Windows [Version 6.0.6000]
Copyright (c) 2006 Microsoft Corporation. All rights reserved.
C:\Users\Administrator>cd C:\Program Files\NetSim Standard\bin
C:\Program Files\NetSim Standard\bin>NetSimCore.exe -apppath "C:\Program Files\NetSim Standard\bin" -iopath C:\Users\ADMINI~1\AppData\Local\Temp\NetSim -license 5053@192.168.0.2

To know more about the options that are available to run NetSim via CLI, type the following in the command prompt.

>cd <app path>

>NetSimCore.exe -h


C:\WINDOWS\system32\cmd.exe
D:\Daisy\AppPath of NetSim\NetSim Standard 7.0.3\bin>NetSimCore.exe -h
Usage: NetSimCore [-AppPath PATH] [-IOPath PATH] [-license Port@IP] [-activate] [-roam flag] [-M flag] [-D] [-h]
-AppPath -- Specify the path where NetSim dll's are present. This is typically the bin folder of the install directory.
-IOPath -- Specify the path where input configuration.xml is present. In the same path output Metrics.txt will be written by NetSim.
-license -- Specify the license server address.
For Floating License this is of the form - port@ipaddress, where port is usually 5053 and ipaddress is the IP address of the PC where the license server is running
For Node locked license (Internet Activated License) this is of the form - .
-activate -- Used to activate sw lock. Netsim will ask key to activate.
-roam -- Used to checkout/checkin the roaming license.
0 to run roaming first(if available without checkout) then floating.
1 to checkout a roaming license.
-1 to checkin a roaming license.
-M -- Memory leak flag /*For Tetcos debug use only*/
2 to dump the memory leak
3 to break at certain allocation number and dump memory leak at end.
Example:
-M 2 dumps the memory leak at the end.
-M 3 Ask for allocation number and dump the memory leak at end.
-D -- Break the simulation at a certain event id
Example:
-D 899 will break the simulation when 899th event is triggered.
-D -1 will only print the line number and file name in event trace
-h -- Display the help

Configuration file

Let's see under the hood to know how NetSim is working.

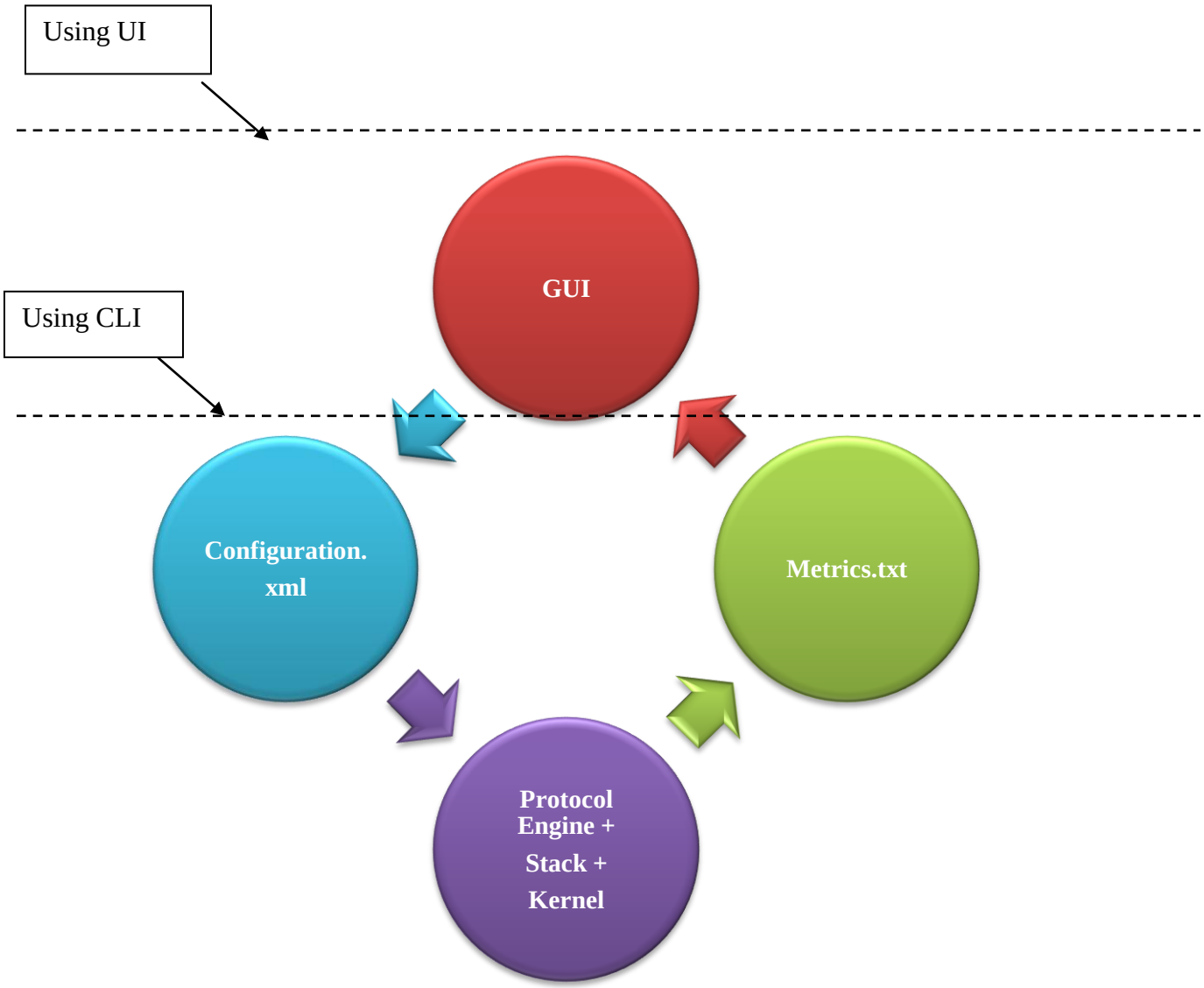


Fig.1. NetSim Architectural Overview

To model a scenario in order to generate metrics in NetSim, GUI will write all the details about the devices used in the scenario and its properties, the links used and their properties, the properties of the environment being used, etc in ***Configuration.xml***.

The back-end engine that contains dlls and NetSimCore.exe will read this Configuration.xml, execute the simulation and write output metrics files (in .txt format) to the IO path. Then, the GUI will display the metrics based on the text files written by the backend.

In order to run NetSim through command line (CLI), the user has to create the Configuration.xml furnishing all the details about the devices, links and the environment of the desired scenario.

How to use Visual Studio to edit the Configuration file?

To edit the Configuration.xml, xml editor is required. There are various xml editors available in the market. Visual Studio 2010 is one of the xml editors that can be used to edit the Configuration.xml with efficacy.

This section shows limelight on how to use Visual Studio to edit the Configuration.xml.

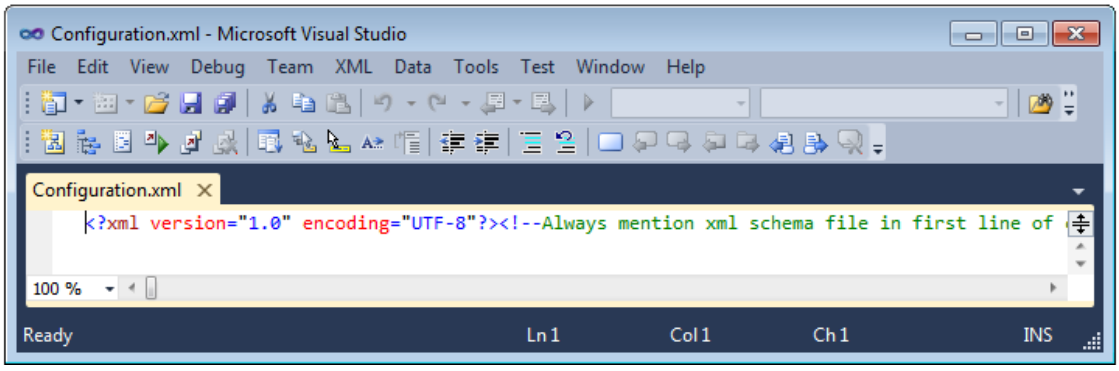
XML View

XML view provides an editor for editing raw XML and provides *IntelliSense* and *color coding*.

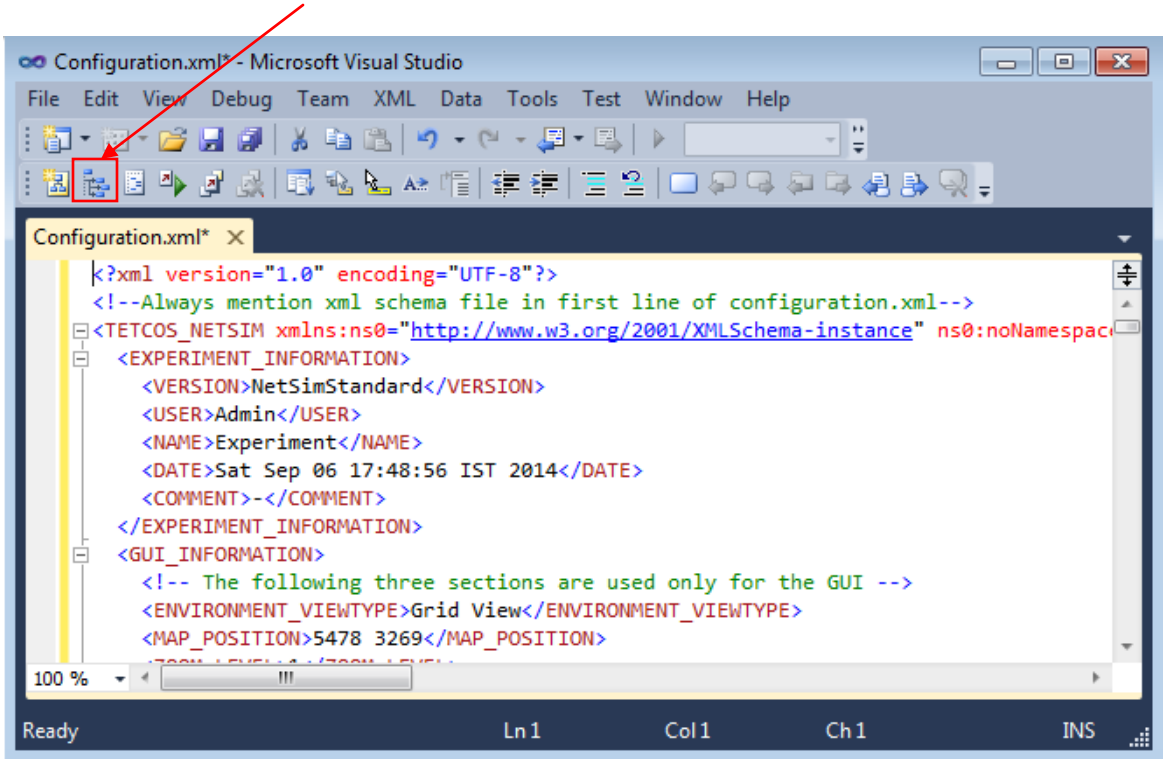
After you type the element name and press the CTRL+ SPACE, you will be presented with a list of attributes that the element supports. This is known as “IntelliSense”. Using this feature, you can select the options that are required to create the desired scenario.

Color coding is followed to indicate the elements and the attributes in a unique fashion.

The following screenshot displays the Configuration.xml which is opened through the Visual Studio.



To reformat it, click on “Reformat Selection” icon.



Sections of Configuration file

These are the different sections in Configuration.xml:

- EXPERIMENT_INFORMATION
- GUI_INFORMATION
- NETWORK_CONFIGURATION
- SIMULATION_PARAMETER
- PROTOCOL_CONFIGURATION
- STATISTICS_COLLECTION

EXPERIMENT_INFORMATION:

This section contains the details about the user credentials, such as the user mode (Admin or Exam or Practice), experiment name, date on which the experiment is created and the comments about the experiment. This section plays a significant role while running NetSim through GUI.

GUI_INFORMATION:

This section contains the GUI information like the environment length, view type etc. and the network name which is desired to be run.

NETWORK_CONFIGURATION:

This section is used to configure the devices and the links of the desired network at the each layer of the OSI stack. It consists of DEVICE_CONFIGURATION, CONNECTION and APPLICATION_CONFIGURATION. DEVICE_CONFIGURATION configures the devices in the desired network while the CONNECTION configures the links in the desired network and APPLICATION configures the Applications.

SIMULATION_PARAMETER:

Simulation time and seed values are described in this section.

PROTOCOL_CONFIGURATION:

IPV4 and static ARP are enabled or disabled in this section. The text files illustrating the static routing and static ARP can be obtained by enabling the corresponding tags in the Configuration.xml.

STATISTICS_COLLECTION:

The packet trace and the event trace can be observed in the text files which are created by enabling the tags in this section. The required fields of the packet trace can be enabled in the PACKET_TRACE while the event trace can be enabled in the EVENT_TRACE of this section.

Troubleshooting

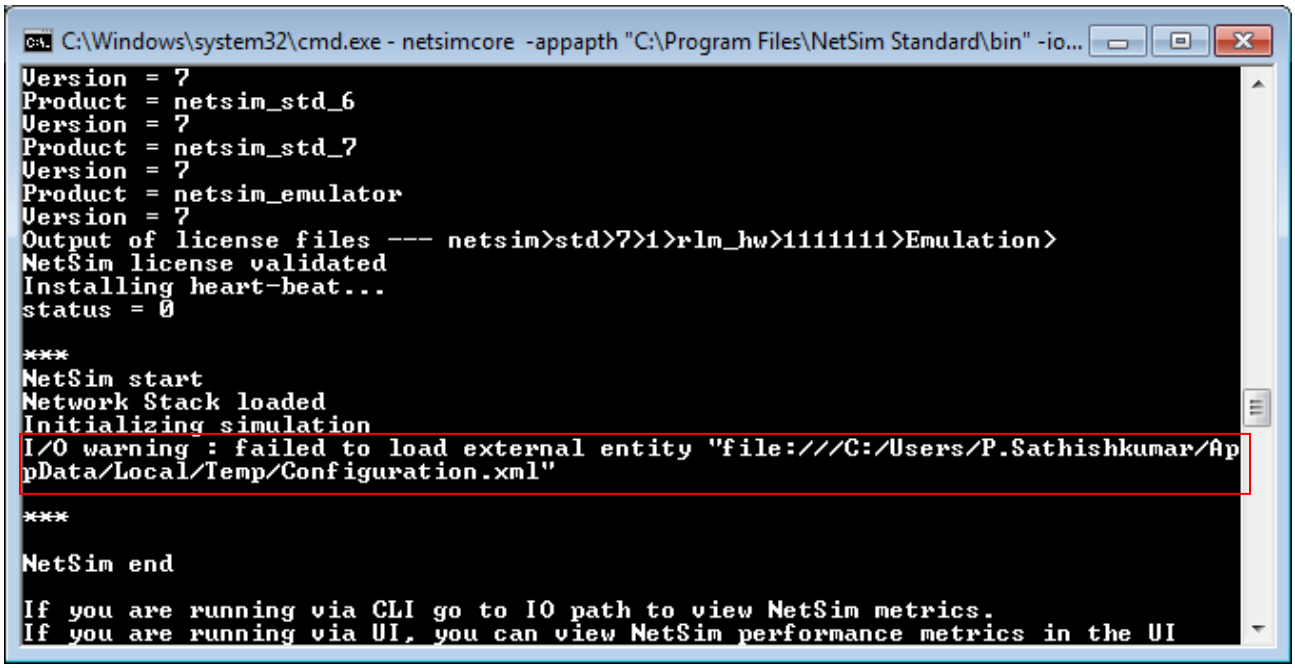
While running NetSim via CLI for the scenarios described in the Configuration file, you may bump into few problems.

> ***Note:*** *While running NetSim via CLI, try to ensure that there are no errors in the Configuration.xml file. The file, ConfigLog.txt, written to the windows temp path would show errors, if any, found by NetSim's config parser.*

This section discusses some common issues and solutions:

Possible errors when running NetSim via CLI

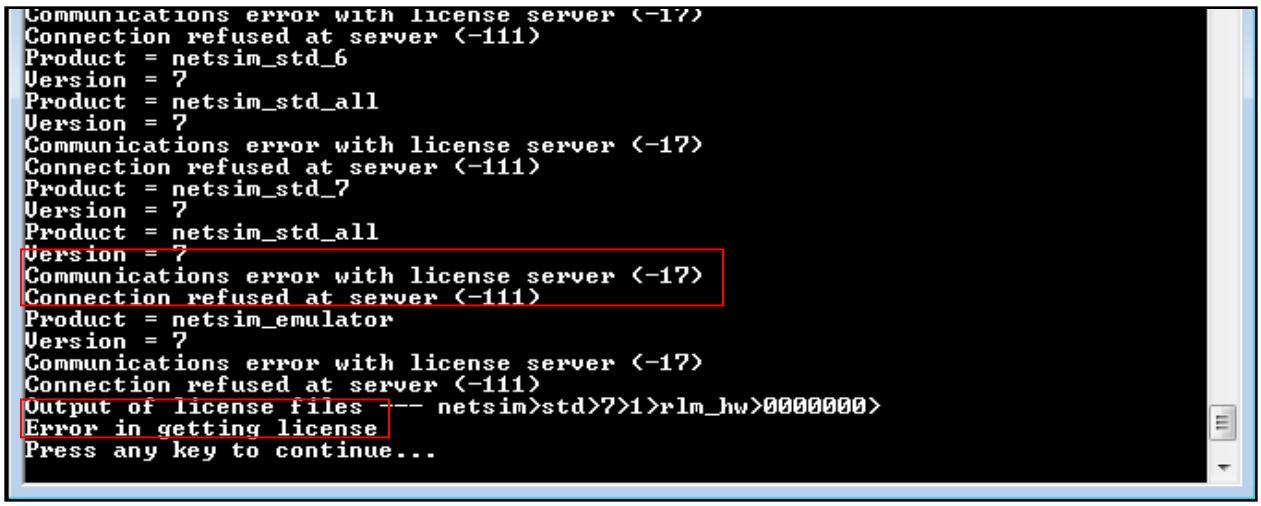
Problem 1: Wrong I/O path


C:\Windows\system32\cmd.exe - netsimcore -appath "C:\Program Files\NetSim Standard\bin" -io...
Version = 7
Product = netsim_std_6
Version = 7
Product = netsim_std_7
Version = 7
Product = netsim_emulator
Version = 7
Output of license files --- netsim>std>7>1>rlm_hw>1111111>Emulation>
NetSim license validated
Installing heart-beat...
status = 0
***
NetSim start
Network Stack loaded
Initializing simulation
I/O warning : failed to load external entity "file:///C:/Users/P.Sathishkumar/Ap
pData/Local/Temp/Configuration.xml"
***
NetSim end
If you are running via CLI go to IO path to view NetSim metrics.
If you are running via UI, you can view NetSim performance metrics in the UI

Solution: While typing the CLI command if you enter wrong I/OPath,or if there is no Configuration.xml file then the following error is thrown. So, check I/O path.

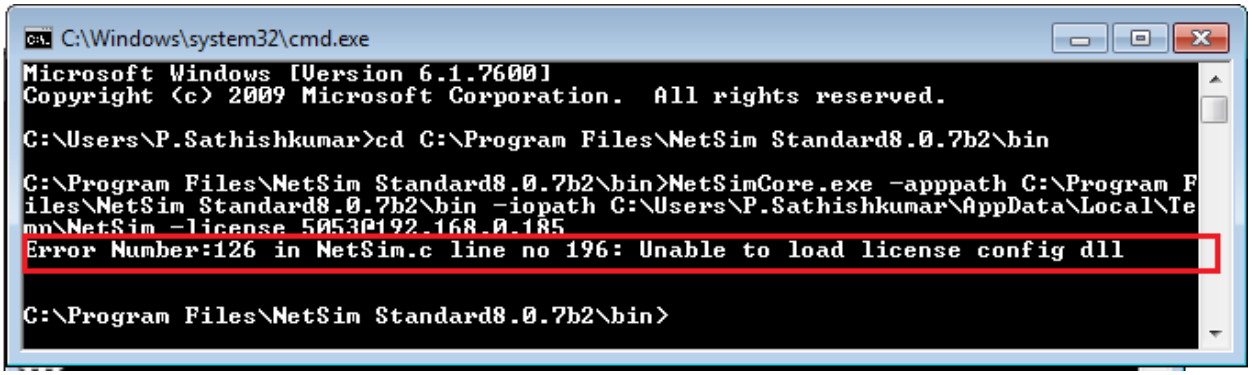
Problem 2: Wrong License information

If the license information is wrong then the following message will be shown.


Communications error with license server (-17)
Connection refused at server (-111)
Product = netsim_std_6
Version = 7
Product = netsim_std_all
Version = 7
Communications error with license server (-17)
Connection refused at server (-111)
Product = netsim_std_7
Version = 7
Product = netsim_std_all
Version = 7
Communications error with license server (-17)
Connection refused at server (-111)
Product = netsim_emulator
Version = 7
Communications error with license server (-17)
Connection refused at server (-111)
Output of license files --- netsim>std>7>1>rlm_hw>0000000>
Error in getting license
Press any key to continue...

Solution: In this example, license server IP address is 192.168.0.185 but it is given as 192.168.0.180. Here server IP address is wrong.Same error message is shown for wrong port number, wrong tag name like–apppath,-iopath,-license. For example, if –appppath is typed instead of –apppath then this message will be shown. So, check those details.

Problem 3: Apppath and I/O path white spaces


C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
C:\Users\P.Sathishkumar>cd C:\Program Files\NetSim Standard8.0.7b2\bin
C:\Program Files\NetSim Standard8.0.7b2\bin>NetSimCore.exe -apppath C:\Program F
iles\NetSim Standard8.0.7b2\bin -iopath C:\Users\P.Sathishkumar\AppData\Local\Te
mp\NetSim -license 5053@192.168.0.185
Error Number:126 in NetSim.c line no 196: Unable to load license config dll
C:\Program Files\NetSim Standard8.0.7b2\bin>

Solution: If the folder name contains white space, then mention the folder path within double quoteswhile specifying the folder name in the command prompt. For example, if app path containswhite space, then the app path must be mentioned within double quotes in the command prompt.

Possible errors in Configuration.xml

Problem 1:*Specific attributes in the Configuration file are highlighted with zigzag lines*

If invalid input is given in the Configuration file, then the corresponding attribute is highlighted in zigzag lines as shown in the figure given below.

```
  <LAYER TYPE="NETWORK_LAYER">
    <!--Set Network_Protocol properties-->
    <!--Configure each layer-->
    <!--Set Physical_Layer properties-->
    <NETWORK_PROTOCOL IP_ADDRESS="192.168.0.4" NAME="IPV4" SUBNI
    <PROTOCOL NAME="ARP" SETPROPERTY="true">
      <PROTOCOL_PROPERTY ARP_RETRY_INTERVAL="10" ARP_RETRY_LIMI'
    </PROTOCOL>
  </LAYER>
</INTERFACE>
<INTERFACE ID="2" INTERFACE_TYPE="ETHERNET">
  <!--Configure each layer-->
  <!--Set Physical_Layer properties-->
  <LAYER TYPE="PHYSICAL_LAYER" CONNECTION_MEDIUM="Wireless">
    <!--Set Protocol properties-->
    <PROTOCOL NAME="ETHERNET" SETPROPERTY="true">
      <!--Set protocol properties-->
      <PROTOCOL_PROPERTY ETHERNET_STANDARD="IEEE802.3u" />
    </PROTOCOL>
  </LAYER>
```

Solution:

To resolve this issue mouse over the corresponding attribute, in order to get the tool tip that furnishes the details about the valid input for that attribute.

> ***Note:*** If the schema file and the configuration file are not present in the same folder, the zigzag lines won't appear. So place the Configuration file and Schema File in the same location or change the path of schema file in the configuration file.

Problem 2:*Simulation runs successfully but errors are provided in the ConfigLog.txt*

Since, NetSim will crash if all the required parameters are not configured, NetSim addresses this issue by "enabling" default values where ever required.

However, the user is made to be aware of the missing parameters in the Configuration file by mentioning the errors in the ConfigLog.txt.

Example:

Error produced in the ConfigLog.txt when the value for the Update timer for router is not specified in the Configuration file is given below.

Attributes "UPDATE_TIMER" not found in line number 349.

Tag path:
/TETCOS_NETSIM/NETWORK_CONFIGURATION/DEVICE_CONFIGURATION/DEVICE [3] /LAYER [2]/ROUTING_PROTOCOL/PROTOCOL_PROPERTY

Solution: User has to specify the missing parameters in the Configuration file which are mentioned in the ConfigLog.txt, in order to obtain accurate metrics for the desired scenario.

Problem 3:*Simulation is interrupted and the error is displayed in command line*

Example: If the user defined function in the PRIMITIVES_INFORMATION of the Configuration file is incorrect/ missing, then the simulation is interrupted and the corresponding function name will be specified in the command prompt as shown below.

In 1330 line of StackInternal.c file following error occurs

fn_NetSim_UDP_Init: The specified procedure could not be found.
Error number=127

Solution: User has to specify the exact function name in the PRIMITIVES_INFORMATION of the Configuration file, in order to simulate the desired scenario.

Problem 4:*Simulation is interrupted and the metrics are not generated*

This issue arises mainly while linking custom code in NetSim. i.e., when the custom code is having any bug, the flow of execution of NetSim gets disturbed and this leads to stand still condition.

Solution: The bugs can be fixed by setting the break points for debugging of custom code and the following steps can be used to fix the issue.

Step 1*:** Since the packets flow per events in NetSim, enable event trace*** eitherin the Configuration file or via the UI. Analyze the Event Trace to identify at which event id the stand still condition occurs.

Step 2: Run NetSim independent of UI by mentioning the following things in the command prompt.

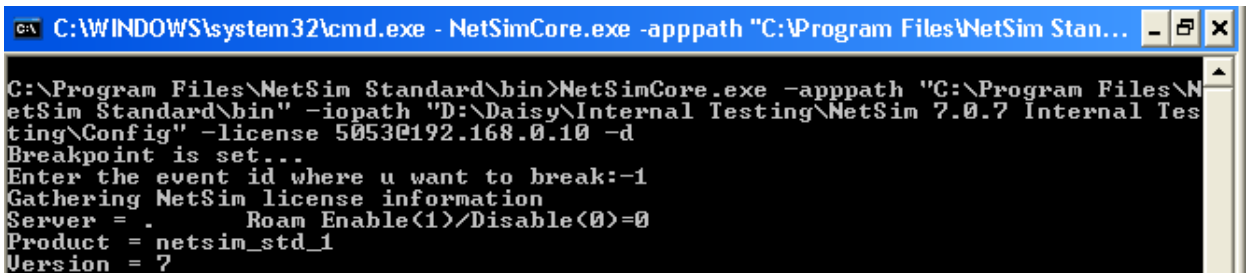
cd<apppath> *-apppath<NetSim install directory> -iopath <IO Path> -license 5053@<License Server IP Address> -d*

Example:

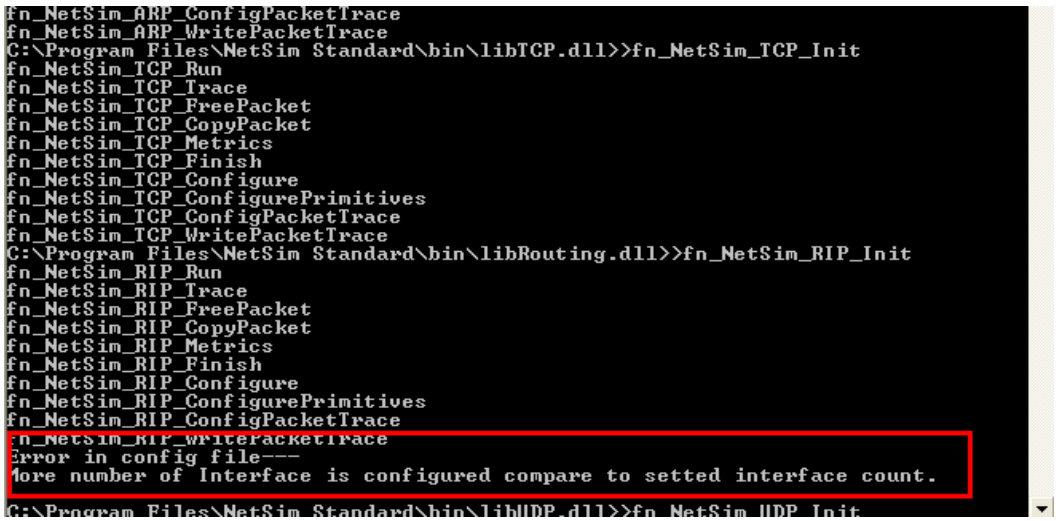
C:\Program Files\NetSim Standard\bin>NetSimCore.exe -apppath "C:\Program Files\NetSim Standard\bin" -iopath "D:\NetSim" -license 5053@192.168.0.10 **-d**

Step 3: Command prompt will ask for break point while debugging.

Set the event id from which the code flow has to be analyzed as the break point in the command prompt as shown below.


C:\WINDOWS\system32\cmd.exe - NetSimCore.exe -apppath "C:\Program Files\NetSim Stan...
C:\Program Files\NetSim Standard\bin>NetSimCore.exe -apppath "C:\Program Files\NetSim Standard\bin" -iopath "D:\Daisy\Internal Testing\NetSim 7.0.7 Internal Testing\Config" -license 5053@192.168.0.10 -d
Breakpoint is set...
Enter the event id where u want to break:-1
Gathering NetSim license information
Server = . Roam Enable(1)/Disable(0)=0
Product = netsim_std_1
Version = 7

Errors will be thrown in the command prompt, if any as shown below.



Step 4:

Fix the error and then run NetSim independent of UI with debug mode as in Step 2. Check whether simulation runs successfully by checking Event Trace, Metrics and ConfigLog files.

> *Note: Set the breakpoint as -1 when the simulation has to run without any pause in any event.*

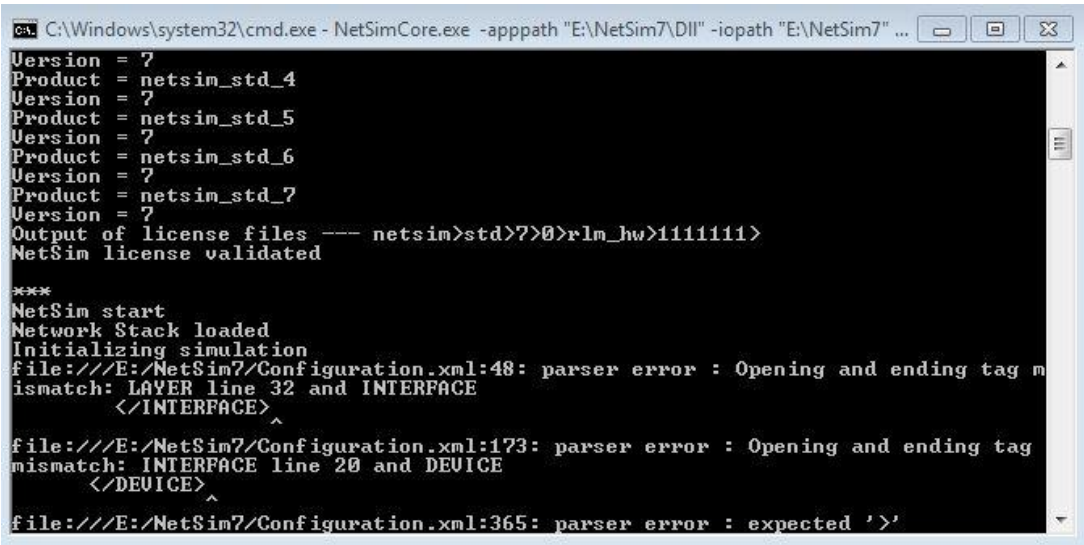
Problem 5: *Simulation does not commence and error is displayed at the command prompt. Also, Zigzag lines appearing at the tag specifying the Layer in the Configuration file*

This issue arises mainly when the closing tag is not specified correctly for a particular layer in the Configuration file.

Example: If the closing tag is not specified for the Data link Layer, then the zigzag lines appear at the starting tags of Data link Layer and the Network Layer.

```
<!--Configure each layer-->
<!--Set DataLink Layer properties-->
<LAYER TYPE="DATALINK_LAYER">
  <!--Set Protocol properties-->
  <PROTOCOL NAME="ETHERNET" SETPROPERTY="true" MAC_ADDRESS="AAAAAAAAAAAA">
    <!--Set Protocol properties-->
    <PROTOCOL_PROPERTY />
  </PROTOCOL>

  <!--Configure each layer-->
<!--Set Network Layer properties-->
<LAYER TYPE="NETWORK_LAYER">
  <!--Set Network Protocol properties-->
  <NETWORK_PROTOCOL IP_ADDRESS="192.168.0.5" NAME="IPV4" SUBNET_MASK="255.255.255.
  <PROTOCOL NAME="ARP" SETPROPERTY="true">
    <PROTOCOL_PROPERTY ARP_RETRY_INTERVAL="10" ARP_RETRY_LIMIT="3" />
  </PROTOCOL>
</LAYER>
</INTERFACE>
```

When NetSim is made to run through CLI, then the following error gets displayed in the command prompt.


C:\Windows\system32\cmd.exe - NetSimCore.exe -apppath "E:\NetSim7\Dll" -iopath "E:\NetSim7" ...
Version = 7
Product = netsim_std_4
Version = 7
Product = netsim_std_5
Version = 7
Product = netsim_std_6
Version = 7
Product = netsim_std_7
Version = 7
Output of license files --- netsim>std>7>0>rlm_hw>1111111>
NetSim license validated
***
NetSim start
Network Stack loaded
Initializing simulation
file:///E:/NetSim7/Configuration.xml:48: parser error : Opening and ending tag m
ismatch: LAYER line 32 and INTERFACE
</INTERFACE>
file:///E:/NetSim7/Configuration.xml:173: parser error : Opening and ending tag
mismatch: INTERFACE line 20 and DEVICE
</DEVICE>
file:///E:/NetSim7/Configuration.xml:365: parser error : expected '>'

Solution: The bug can be fixed by setting the closing tag correctly in the Configuration file

Problem 6: *Zigzag lines appearing at configuration.xsd in the Configuration file*

This issue arises when the schema and the configuration file are not in the same folder.

```
--Config file-->
--Always mention xml schema file in first line of configuration.xml-->
ETCOS_NETSIM xmlns:ns0="http://www.w3.org/2001/XMLSchema-instance" ns0:noNamespaceSchemaLocation="Configuration.xsd
<EXPERIMENT_INFORMATION>
  <VERSION>NetSim Standard 7.0</VERSION>
  <USER>Admin</USER>
  <NAME>NetSim</NAME>
  <DATE>Thu Nov 05 14:37:02 IST 2012</DATE>
  <COMMENT>NetSim Configuration File</COMMENT>
</EXPERIMENT_INFORMATION>
<NETWORK_CONFIGURATION>
  <!--The following section of config file is used to set the device properties-->
  <DEVICE_CONFIGURATION DEVICE_COUNT="8">
    <!--Set Device properties-->
    <DEVICE DEVICE_NAME="MS1" DEVICE_ID="1" INTERFACE_COUNT="1" TYPE="MOBILE_STATION">

        <!--Set Pos_3D properties-->
        <POS_3D X-POS="157" Y-POS="110" Z-POS="0" />
        <!--Set Interface properties-->
        <INTERFACE ID="1" INTERFACE_TYPE="CDMA">
          <!--Configure each layer-->
          <!--Set Physical_Layer properties-->
          <LAYER TYPE="PHYSICAL_LAYER" CONNECTION_MEDIUM="WIRELESS">
            <!--Set Protocol properties-->
            <PROTOCOL NAME="CDMA"  SETPROPERTY="true">
              <!--Set Protocol properties-->
              <!--
      Unit-
        TRANSMITTED POWER = W
```

Solution: The bug can be fixed by placing the Configuration file and schema in the same folder.

Problem 7:*Simulation does not commence and the error is displayed at the command prompt*

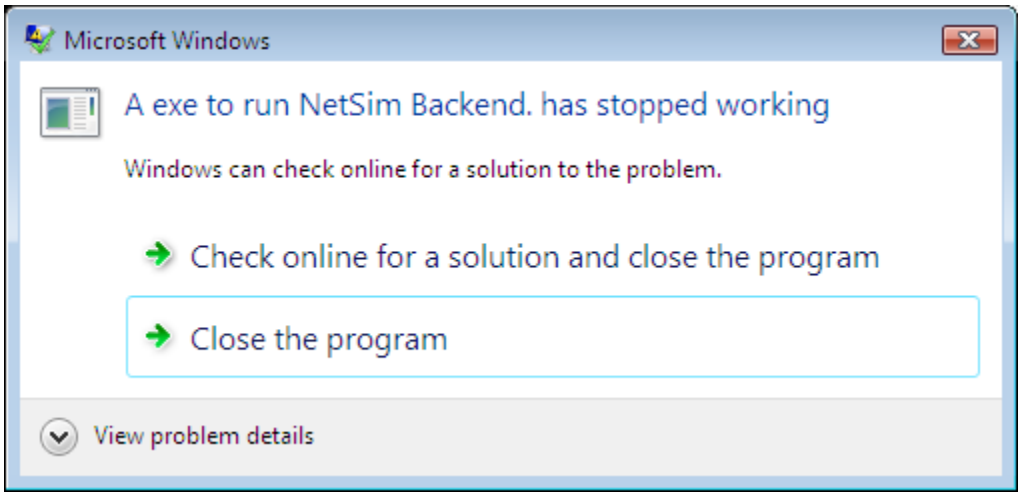
Example:

This problem arises when the Configuration.xml (file name and extension should be correct) is not available in the specified io path.

Solution: The bug can be fixed by ensuring whether file name and extension of Configuration.xml are correct.

Problem 8: *Simulation terminates and exhibits unpredictable behavior. An error message stating, "An exe to run NetSim backend has stopped working" is thrown*

Example:


Microsoft Windows
A exe to run NetSim Backend. has stopped working
Windows can check online for a solution to the problem.
Check online for a solution and close the program
Close the program
View problem details

This problem arises if there is any flaw in the Configuration.xml or in the dll.

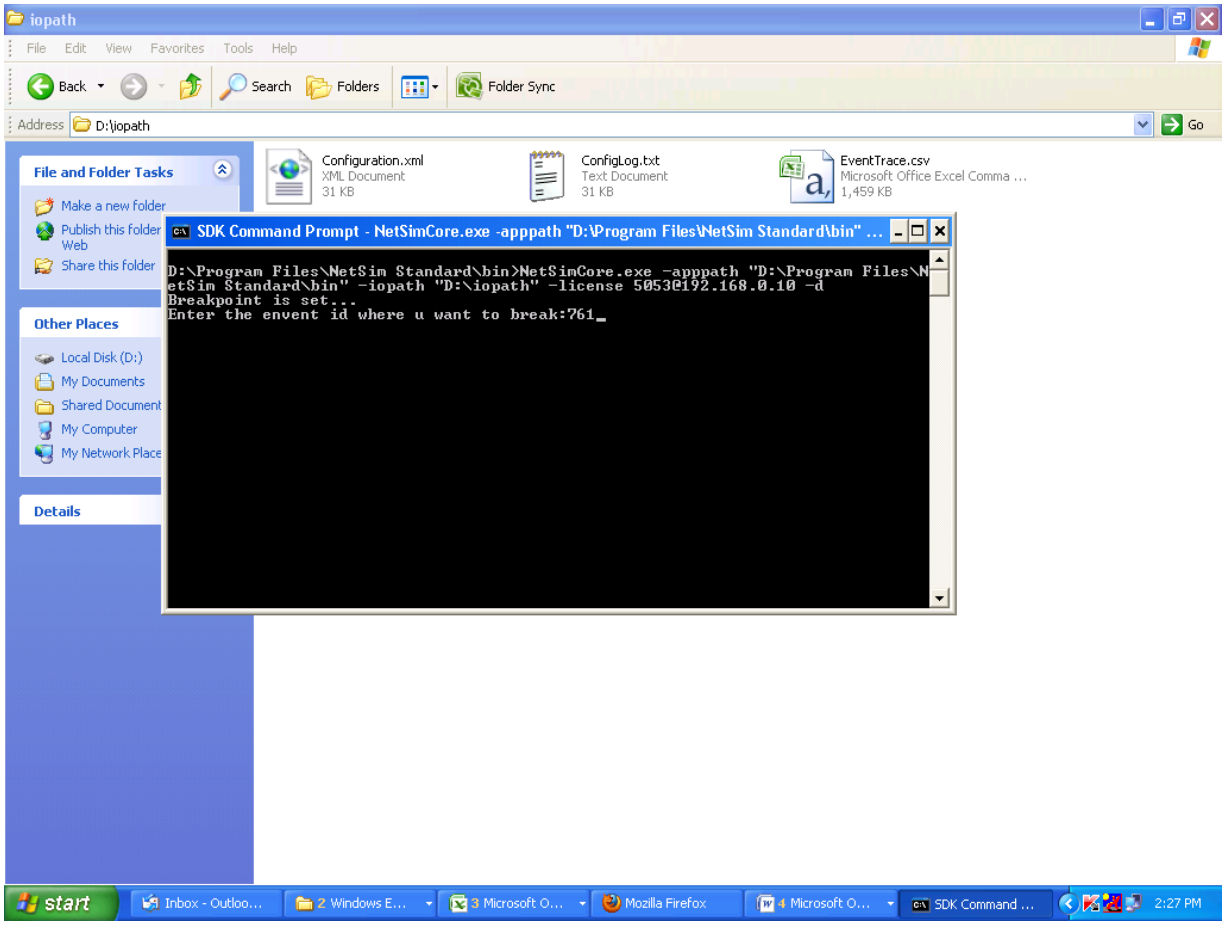
Solution: Check whether the desired scenario has been configured properly in the Configuration.xml.

Enable the event trace in the Configuration.xml so that the event details can be logged.

Run NetSim via CLI again by specifying the event ID in order to debug the code, as given below.

>NetSimCore.exe<space> -apppath<space><app path><space>-iopath<space><io path><space>-license<space>5053@<Server IP Address><space> -d

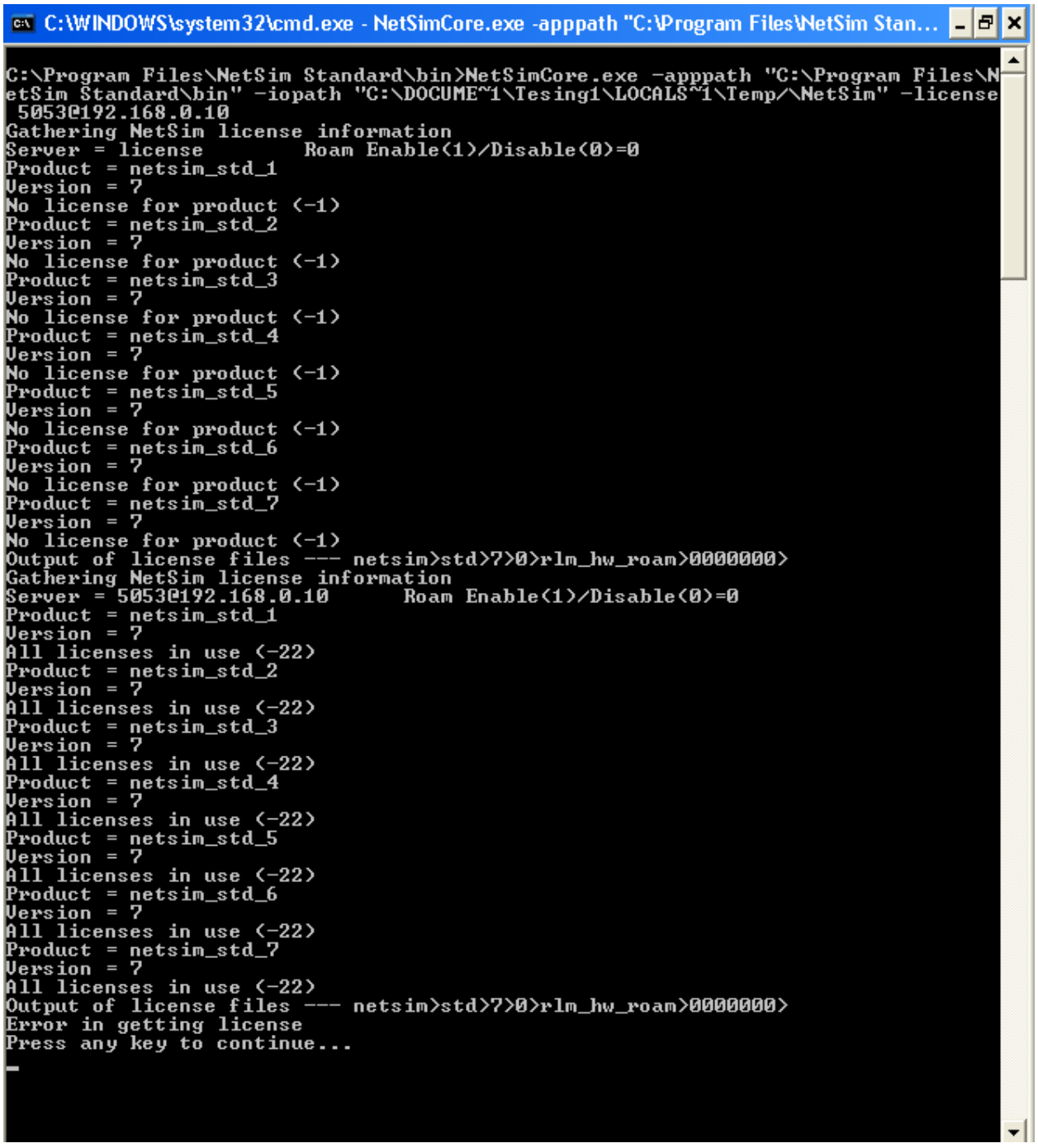
Event id will be asked in the command prompt. Enter the event id from where the code has to be analyzed, as given below.



Now, user can debug the code, so that the flaw in the Configuration.xml or in the dll can be removed, easily.

Problem 9: *Simulation does not commence."No license for product (-1)" is displayed in the command prompt.*

Example:


C:\WINDOWS\system32\cmd.exe - NetSimCore.exe -apppath "C:\Program Files\NetSim Stan...
C:\Program Files\NetSim Standard\bin>NetSimCore.exe -apppath "C:\Program Files\NetSim Standard\bin" -iopath "C:\DOCUME~1\Tesing1\LOCALS~1\Temp/\NetSim" -license 5053@192.168.0.10
Gathering NetSim license information
Server = license Roam Enable(1)/Disable(0)=0
Product = netsim_std_1
Version = 7
No license for product (-1)
Product = netsim_std_2
Version = 7
No license for product (-1)
Product = netsim_std_3
Version = 7
No license for product (-1)
Product = netsim_std_4
Version = 7
No license for product (-1)
Product = netsim_std_5
Version = 7
No license for product (-1)
Product = netsim_std_6
Version = 7
No license for product (-1)
Product = netsim_std_7
Version = 7
No license for product (-1)
Output of license files --- netsim>std>7>0>rlm_hw_roam>0000000>
Gathering NetSim license information
Server = 5053@192.168.0.10 Roam Enable(1)/Disable(0)=0
Product = netsim_std_1
Version = 7
All licenses in use (-22)
Product = netsim_std_2
Version = 7
All licenses in use (-22)
Product = netsim_std_3
Version = 7
All licenses in use (-22)
Product = netsim_std_4
Version = 7
All licenses in use (-22)
Product = netsim_std_5
Version = 7
All licenses in use (-22)
Product = netsim_std_6
Version = 7
All licenses in use (-22)
Product = netsim_std_7
Version = 7
All licenses in use (-22)
Output of license files --- netsim>std>7>0>rlm_hw_roam>0000000>
Error in getting license
Press any key to continue...

Solution:

NetSim is based on the client-server architecture. When NetSim runs in the client machine, it will check for the license in the same machine, first. If license is not available in the same machine, then “No license for product (-1)” will be displayed in the command prompt and the server machine will be checked for the availability of license. If no license is available in the server machine also, then again “No license for product (-1)” will be displayed in the command prompt.

So, if ”No license for product(-1)” is displayed in the command prompt two times, then check in the NetSim license server to know about the availability of license and adjust the number of current users of NetSim, in order to get the license.

How to write Custom Code in NetSim?

NetSim allows the user to write the custom code for all the protocols by creating a DLL (Dynamic Link Library) for their custom code and replacing NetSim's default DLL with the user's custom DLL.

There are two important steps in this process, and each of these steps has various options as explained in the subsequent pages:

1. Part 1 -- Building your own DLLs
2. Part 2 – Linking your custom DLL with NetSim

> ***Note:*** *Writing custom code and Configuration file serve unique purposes. Configuration file is used to create a scenario (when users want to run NetSim via CLI) whereas, writing custom code is to modify protocol source code and link them as DLLs.*

Dynamic Link Library (DLL)

DLL is the shared library concept, implemented by Microsoft. All DLL files have a .dll file extension. DLLs provide a mechanism for sharing code and data to upgrade functionality without requiring applications to be re-linked or re-compiled. It is not possible to directly execute a DLL, since it requires an EXE for the operating system to load it through an entry point.

Part 1: How to build Dynamic Link Libraries (DLLs)?

> ***Note:*** *If the DLL needs to be debugged then build the DLL in "debug" mode or else build in "release" mode.*

Dynamic link libraries (DLLs) can be built using different compilers:

(a) Using Microsoft Visual Studio Compiler

a) Compiler: Microsoft Visual Studio

To create the DLL follow the steps given below.

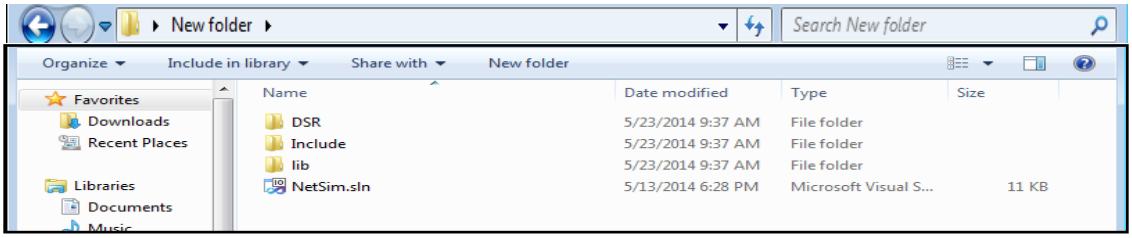
> ***Note:*** *Make sure that visual studio has been installed in your computer*

Step 1: Copy the source code folder of the protocol library which you wish to change. This would be present in "..NetSim_Standard/src/simulation" folder and paste it in a suitable location, for example a new folder on your desktop. For example, if you wish to modify the DSR protocol then copy paste the folder "..NetSim_Standard/src/simulation/DSR" to a new folder in the desktop.

Note: *It is recommended to keep a separate copy of the ../NetSim Standard/src/Simulation folder to have the backup of the original source code.*

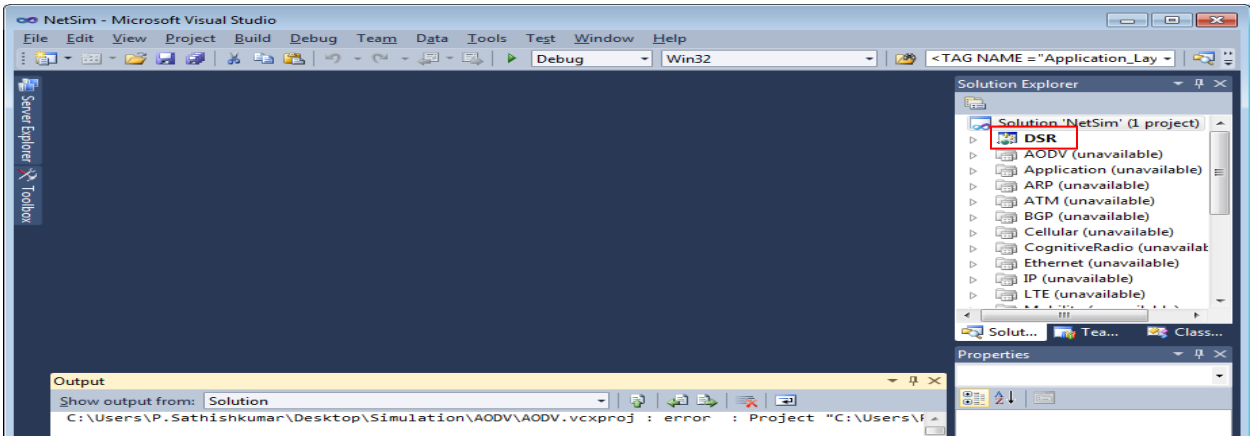
Step 2: Inside this new folder on your desktop also copy and paste a) Include folder b) lib folder and c) NetSim.sln file from "..NetSim_Standard/src/simulation" folder.

For example, if you are modifying the DSR protocol, then the new folder on your desktop would have

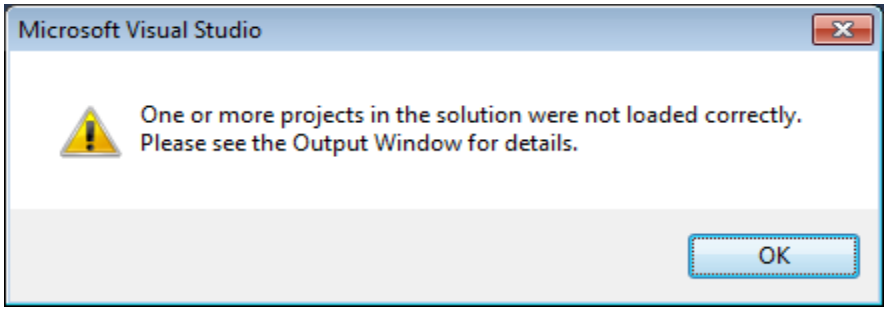


Note: *NetSim allows you to build DLLs for more than one protocol at a time. For example, if you want to build DLLs for DSR and AODV copy and paste both AODV and DSR folders along with,* ***Include, lib and NetSim.sln*** from the simulation folder**.**

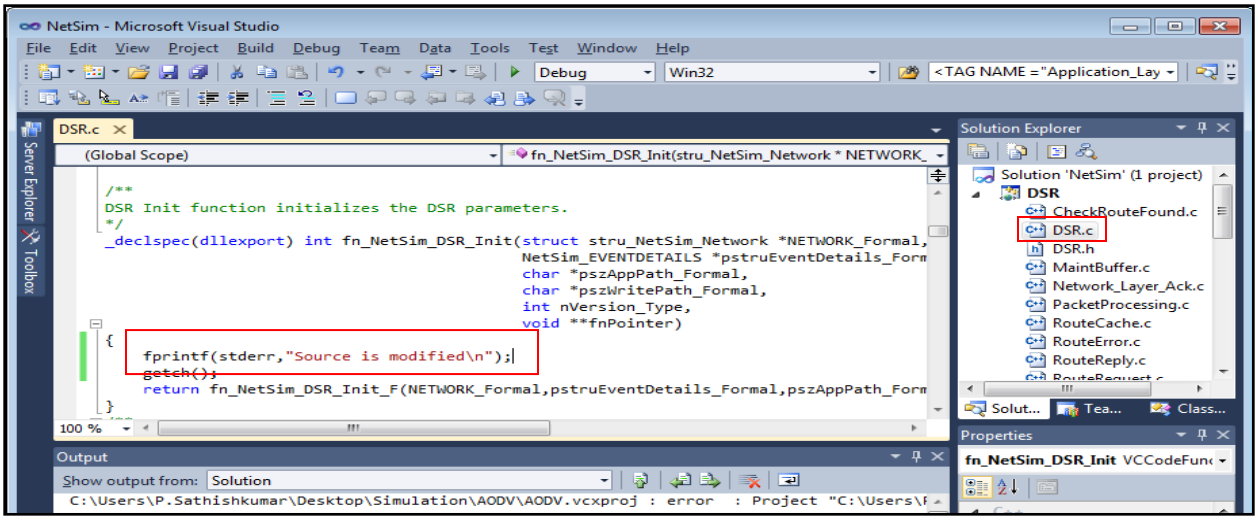
Double click on "NetSim.sln" file to open the solution in visual studio.



Note: *Sometimes you may get the following warning message. Then click on ok and proceed.*

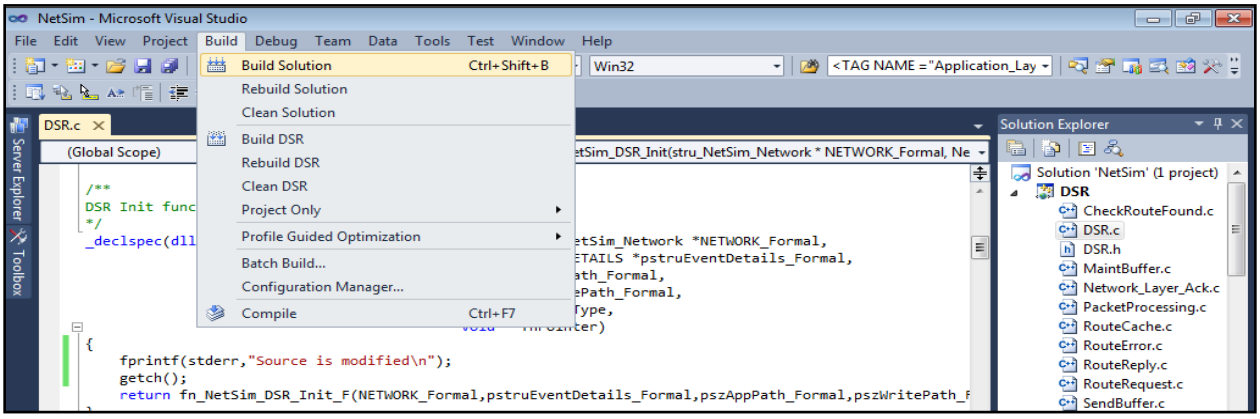

Microsoft Visual Studio
One or more projects in the solution were not loaded correctly.
Please see the Output Window for details.
OK

Select the appropriate protocol and modify the necessary C files inside it.

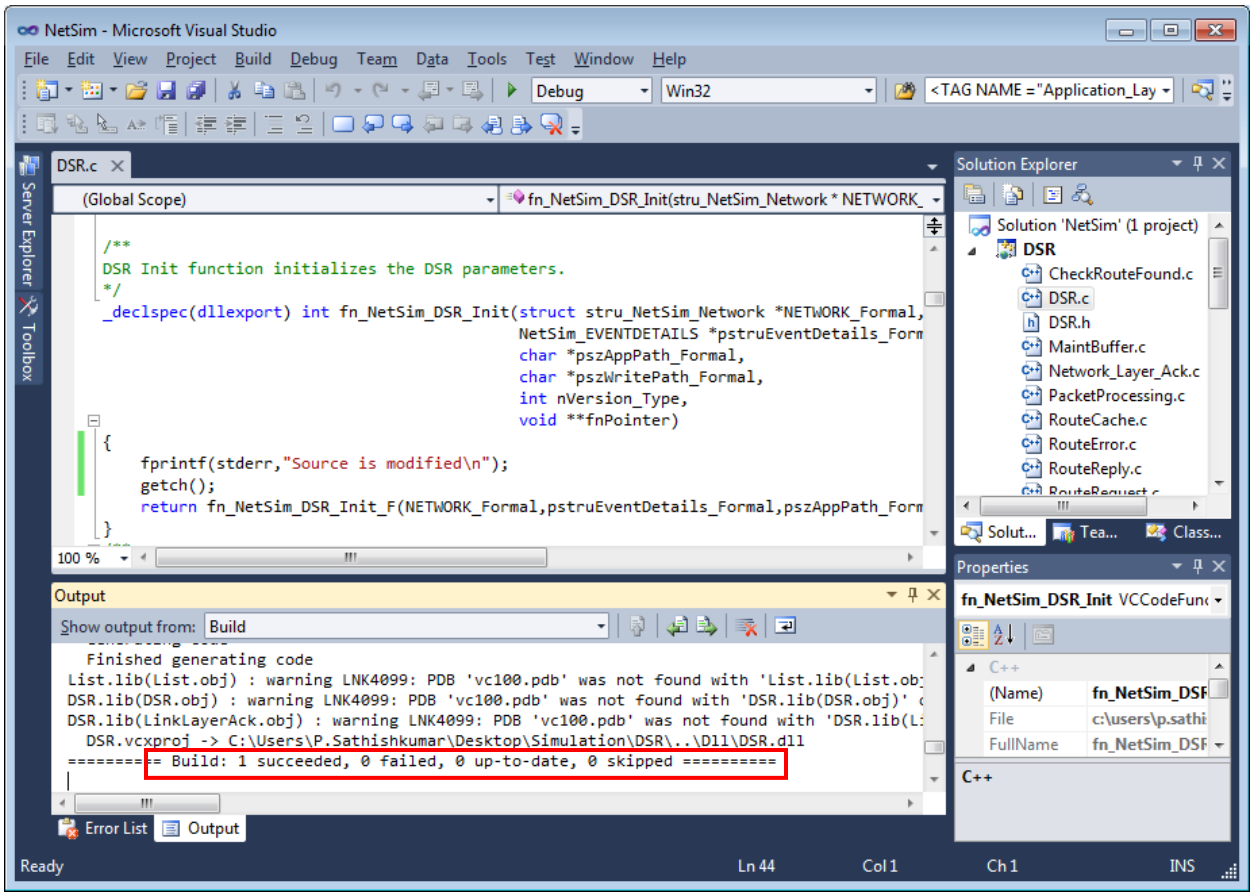


Once this is done click to save the changes and overwrite the file (in case of write protection).

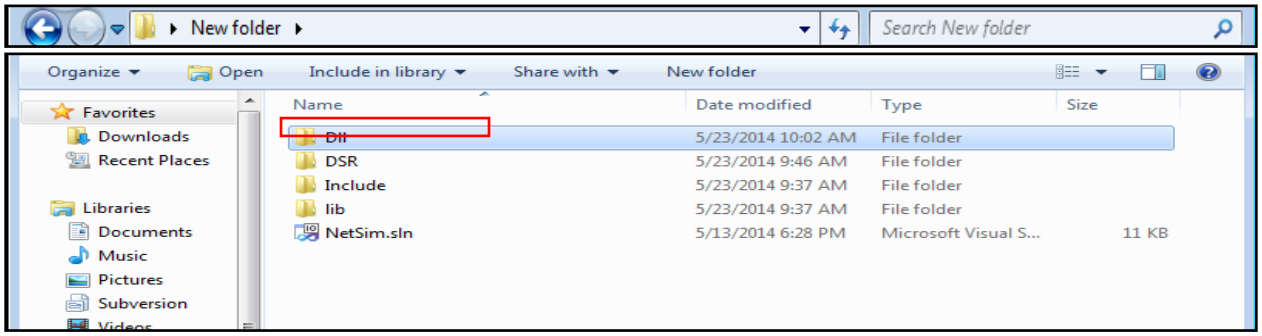
Step 3: Click on "Build->Build Solution"

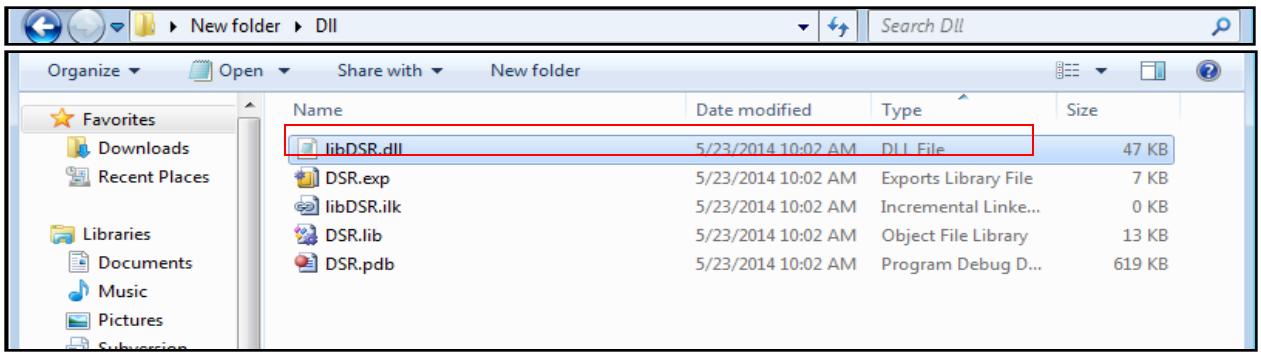


Check that the build is successful and necessary DLL files have been built.



Step 4: Inside the simulation folder (present on the desktop) a Dll folder will be created and inside the DLL's are built.





These is (are) the custom DLL's which have been built per the modifications done. These DLL(s) will have to be pasted in "..NetSim_Standard/bin" path in order to simulate your custom code.

Part 2: How to link dynamic link libraries (DLL)

The procedure to link custom dynamic link libraries (DLLs) is explained below with an example. This procedure is common for any linking of any custom DLL to NetSim.

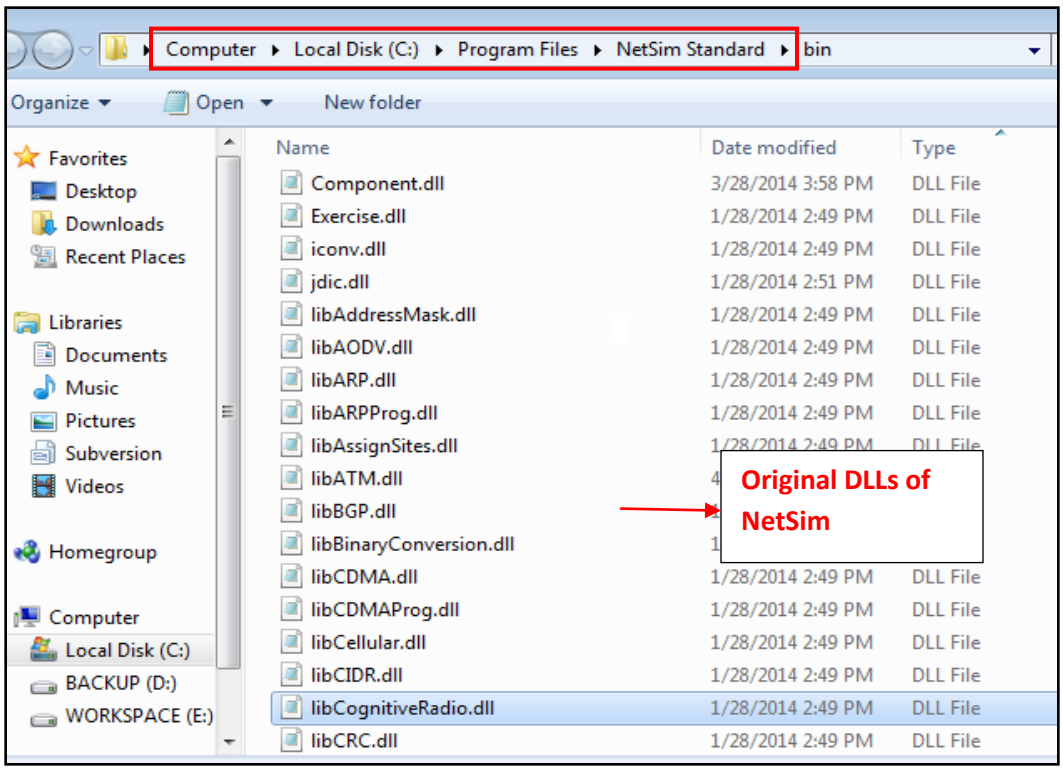
Let us take an example where a user wishes to link their customized CognitiveRadio.dll to the NetSim.

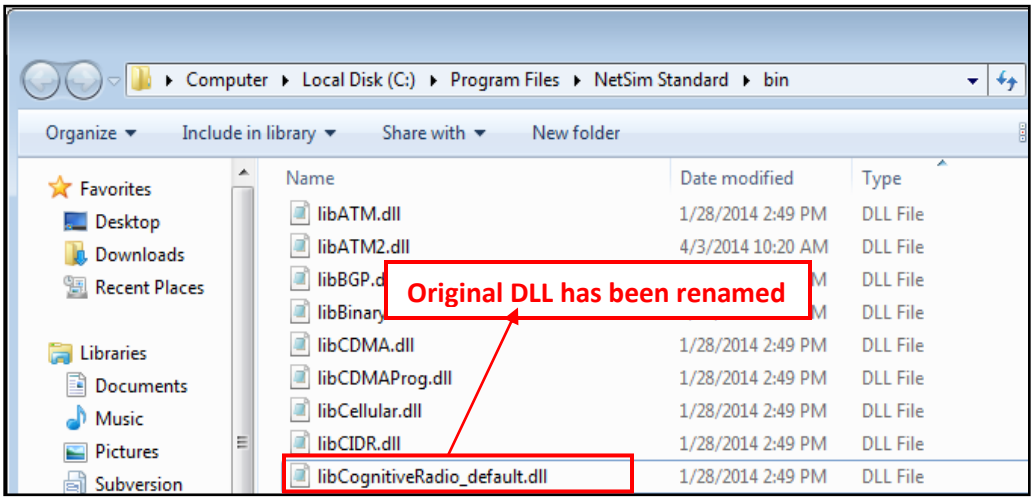
Step 1: Build your own DLL with the source code available in "NetSim Standard->src -> Simulation" folder (for help refer How to build Dynamic Link Libraries section).

Step 2: Go to "NetSim Standard->bin" folder. Here one would find default DLLs for respective protocols as shown in next figure.

Step 3: Now the user must build and place their DLL (custom DLL) inside this folder. First rename then place like this:

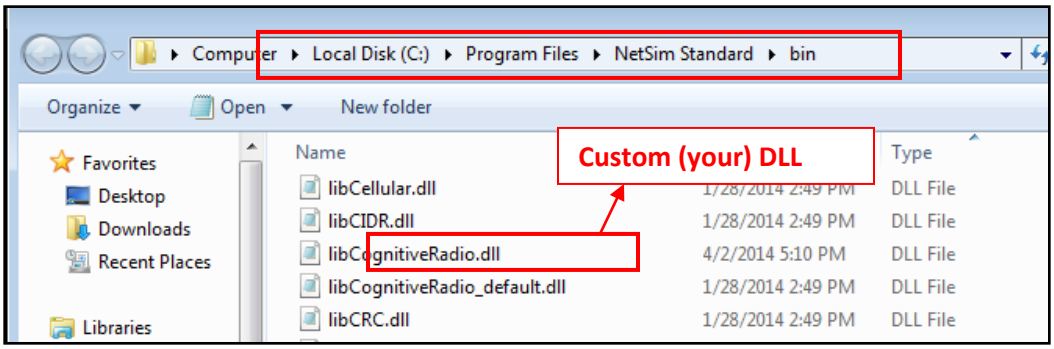
1. **Rename the default DLL**
   It is recommended to rename the default DLL thereby saving a copy of original DLL. For example libCognitiveRadio.dll has been renamed as libCognitiveRadio_default.dll as shown below.





2. **Place the custom DLL inside of the "NetSim Standard->bin" folder.**
   Copy the built custom DLL and paste it inside of the "NetSim Standard->bin" folder.

***Note**: - Custom DLL is the DLL which is created by the user. To create the custom DLL see the help in "NetSim Help->Writing Custom Code in NetSim->How to create dynamic link libraries (dlls)" section.*



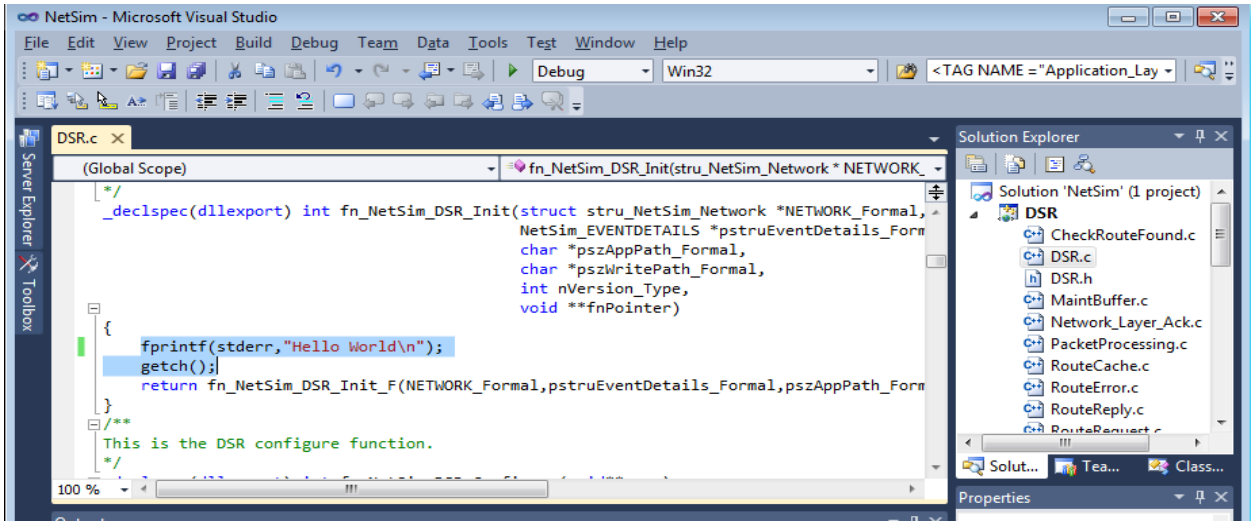
***Note**: Ensure that the name of the DLL which is pasted is exactly the same as the original DLL (before it was renamed). For the example case of cognitive radio, the DLL must be libCognitiveRadio.dll.*

NetSim -- “Hello World Program”

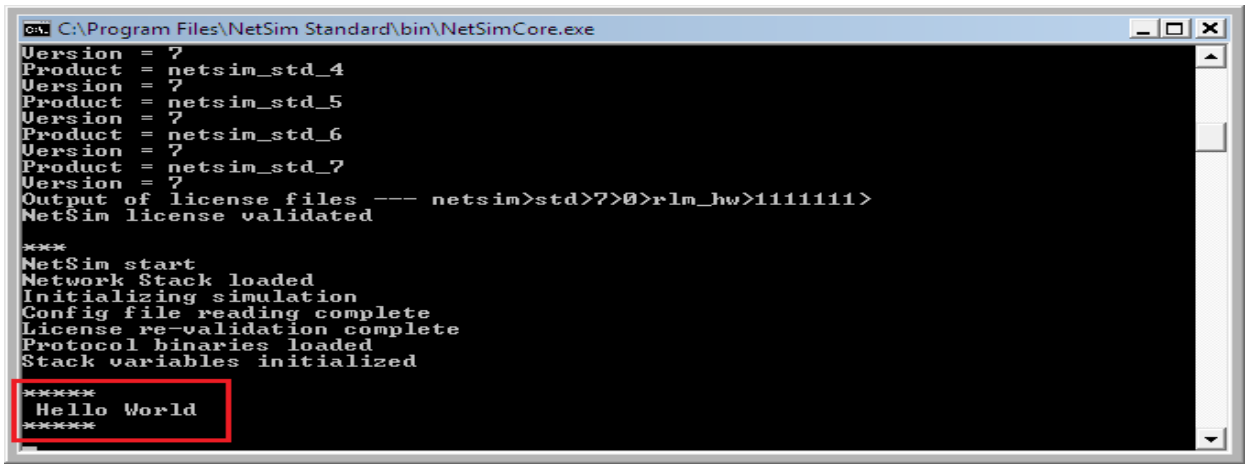
Add a printf statement inside the source code to print “Hello World” when custom built dll is executing.

> ***Note: -*** *stdout is redirected to LogFile.txt which is present in <iopath>. So you can see “Hello World” Statement (or whatever you write inside the printf) inside the LogFile.txt. Use fprintf (stderr,”Hello World”) instead of printf ( ) to write to standard out.*

Take source code of any primitive, for example DSR. Add printf statement as shown below.



Create DLL and Link the DLL to the NetSim explained above. And run the simulation, you can see the following output on the console.



Press enter then simulation will continue.

Debugging custom code in NetSim

Step 1:- Open the Command prompt.

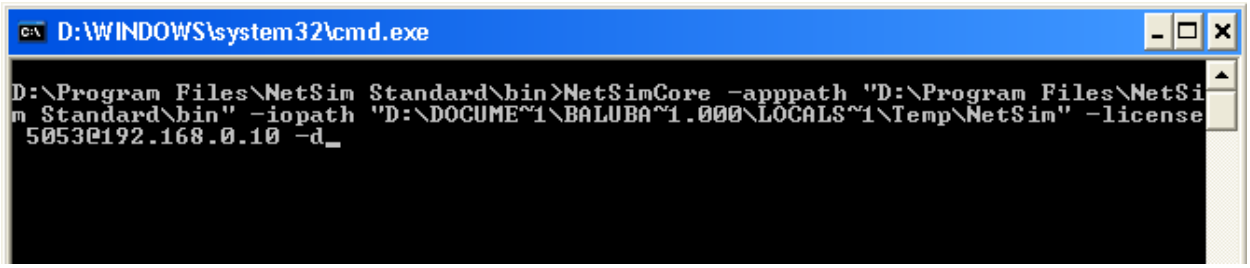
Press “windows+R” and type “cmd”.

Step 2:- To run the NetSim via CLI copy the path where “NetSimCore.exe “ is present.

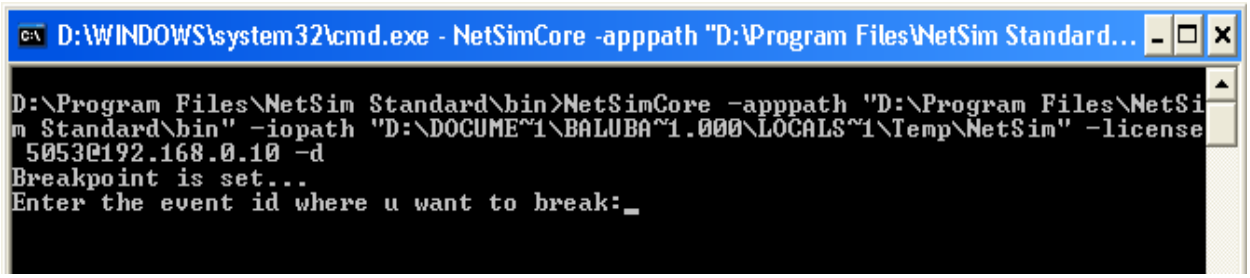
```
>cd <apppath>
```

Step 3:- Type the following command.

```
>NetSimCore.exe<space> -apppath<space><app path><space>-iopath<space><io path><space>-license<space>5053@<Server IP Address> -d
```

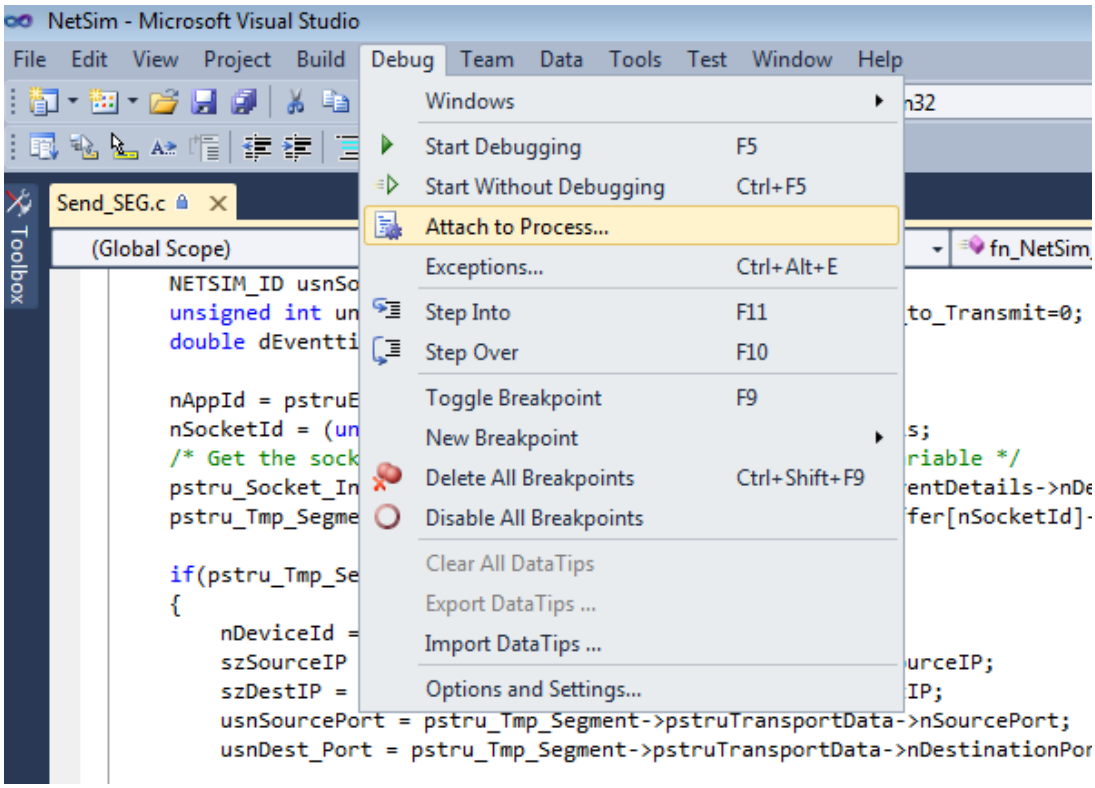

D:\WINDOWS\system32\cmd.exe
D:\Program Files\NetSim Standard\bin>NetSimCore -apppath "D:\Program Files\NetSim Standard\bin" -iopath "D:\DOCUME~1\BALUBA~1.000\LOCALS~1\Temp\NetSim" -license 5053@192.168.0.10 -d

Press enter. Now you can see the following screen.

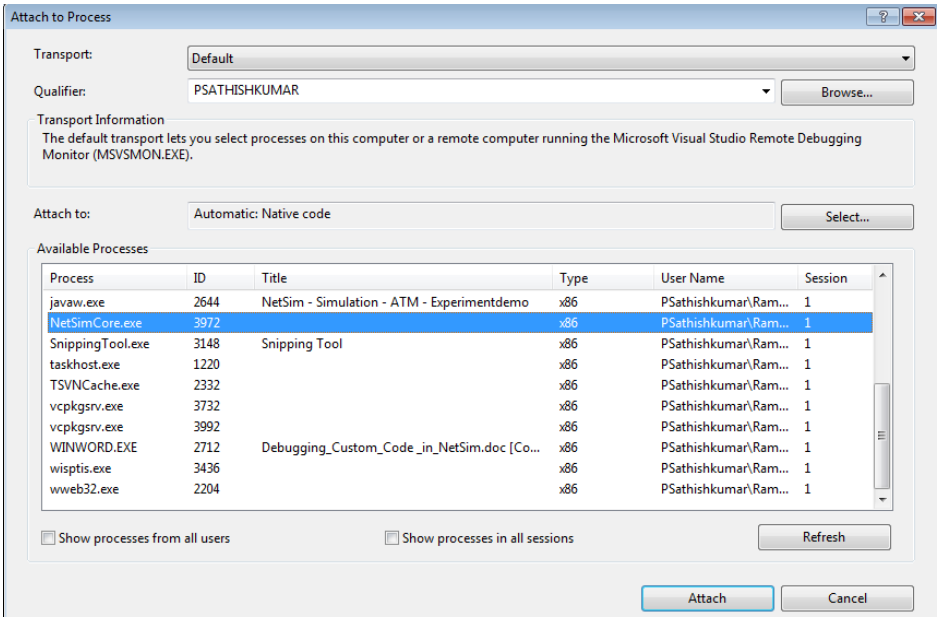

D:\WINDOWS\system32\cmd.exe - NetSimCore -apppath "D:\Program Files\NetSim Standard...
D:\Program Files\NetSim Standard\bin>NetSimCore -apppath "D:\Program Files\NetSim Standard\bin" -iopath "D:\DOCUME~1\BALUBA~1.000\LOCALS~1\Temp\NetSim" -license 5053@192.168.0.10 -d
Breakpoint is set...
Enter the event id where u want to break:

Step 4:- Open/Create the Project in Visual Studio and put break point inside the source code.

Step 5:- Go to “Debug->Attach to Process”



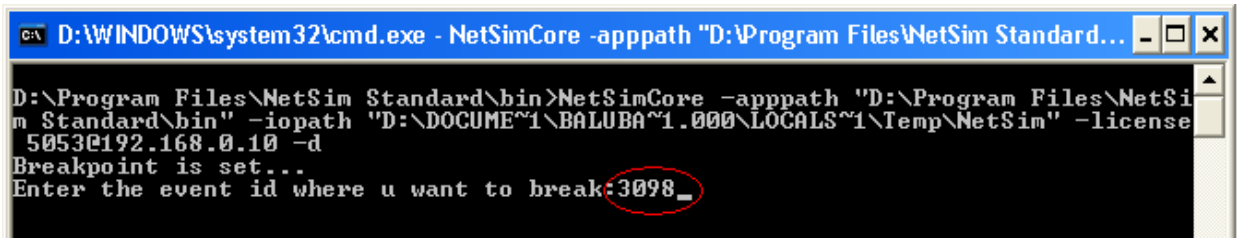
Attach to NetSimCore.exe.


Attach to Process
Transport: Default
Qualifier: PSATHISHKUMAR
Browse...
Transport Information
The default transport lets you select processes on this computer or a remote computer running the Microsoft Visual Studio Remote Debugging Monitor (MSVSMON.EXE).
Attach to: Automatic: Native code
Select...
Available Processes
Process ID Title Type User Name Session
javaw.exe 2644 NetSim - Simulation - ATM - Experimentdemo x86 PSathishkumar\Ram... 1
NetSimCore.exe 3972 x86 PSathishkumar\Ram... 1
SnippingTool.exe 3148 Snipping Tool x86 PSathishkumar\Ram... 1
taskhost.exe 1220 x86 PSathishkumar\Ram... 1
TSVNCache.exe 2332 x86 PSathishkumar\Ram... 1
vcpkgsrv.exe 3732 x86 PSathishkumar\Ram... 1
vcpkgsrv.exe 3992 x86 PSathishkumar\Ram... 1
WINWORD.EXE 2712 Debugging_Custom_Code _in_NetSim.doc [Co... x86 PSathishkumar\Ram... 1
wisptis.exe 3436 x86 PSathishkumar\Ram... 1
wwweb32.exe 2204 x86 PSathishkumar\Ram... 1
Show processes from all users
Show processes in all sessions
Refresh
Attach
Cancel

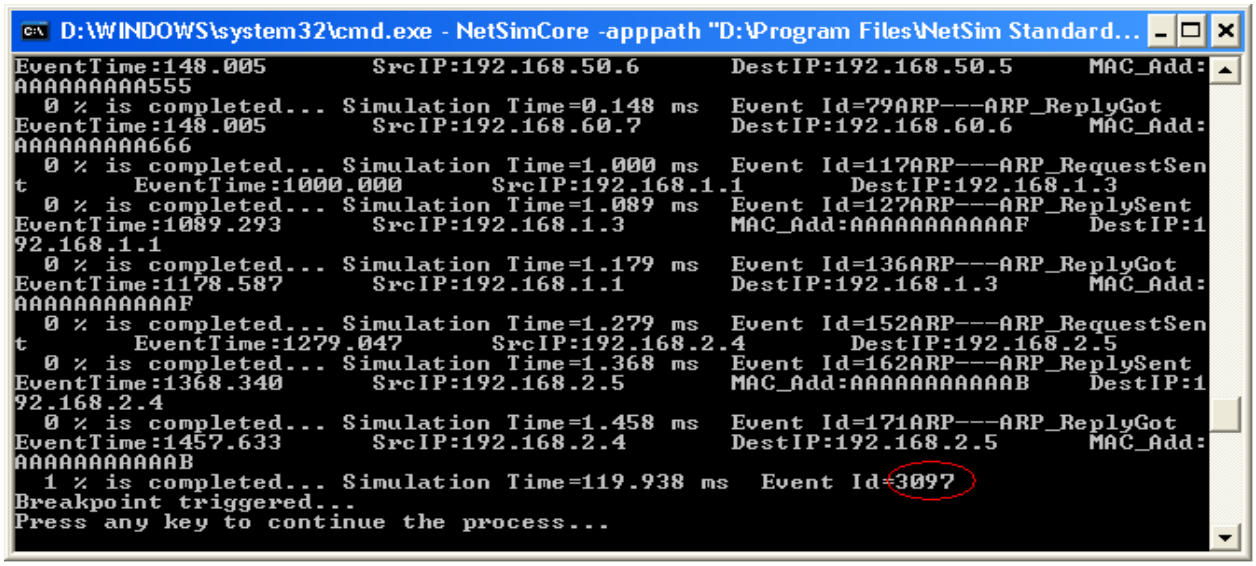
Click on Attach.

Step 6:- Go to command prompt which is already opened in Step 3. Enter the Event Id.

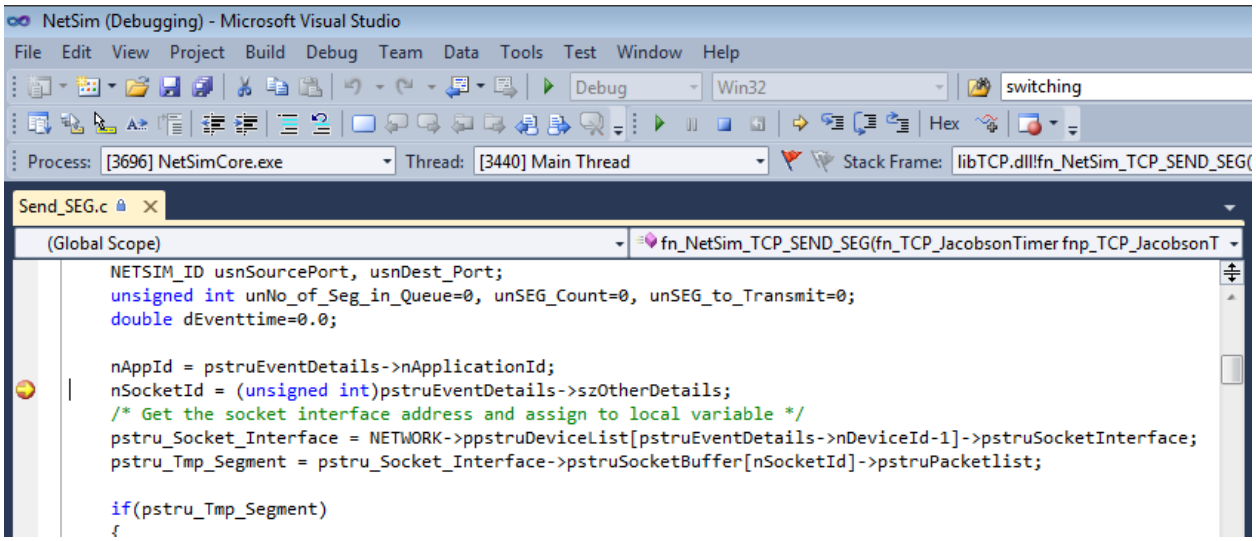
Note:*For help about event details see "Help->NetSim Help F1->Generating Event trace".*
Note:*If you don't want to stop at any event you can specify 0 as event id.*


D:\WINDOWS\system32\cmd.exe - NetSimCore -apppath "D:\Program Files\NetSim Standard...
D:\Program Files\NetSim Standard\bin>NetSimCore -apppath "D:\Program Files\NetSi
m Standard\bin" -iopath "D:\DOCUME~1\BALUBA~1.000\LOCALS~1\Temp\NetSim" -license
5053@192.168.0.10 -d
Breakpoint is set...
Enter the event id where u want to break:3098

Execution will stop at the specified event.


D:\WINDOWS\system32\cmd.exe - NetSimCore -apppath "D:\Program Files\NetSim Standard...
EventTime:148.005 SrcIP:192.168.50.6 DestIP:192.168.50.5 MAC_Add:
AAAAAAAAA555
0 % is completed... Simulation Time=0.148 ms Event Id=79ARP---ARP_ReplyGot
EventTime:148.005 SrcIP:192.168.60.7 DestIP:192.168.60.6 MAC_Add:
AAAAAAAAA666
0 % is completed... Simulation Time=1.000 ms Event Id=117ARP---ARP_RequestSen
t EventTime:1000.000 SrcIP:192.168.1.1 DestIP:192.168.1.3
0 % is completed... Simulation Time=1.089 ms Event Id=127ARP---ARP_ReplySent
EventTime:1089.293 SrcIP:192.168.1.3 MAC_Add:AAAAAAAAAAAF DestIP:1
92.168.1.1
0 % is completed... Simulation Time=1.179 ms Event Id=136ARP---ARP_ReplyGot
EventTime:1178.587 SrcIP:192.168.1.1 DestIP:192.168.1.3 MAC_Add:
AAAAAAAAAAAF
0 % is completed... Simulation Time=1.279 ms Event Id=152ARP---ARP_RequestSen
t EventTime:1279.047 SrcIP:192.168.2.4 DestIP:192.168.2.5
0 % is completed... Simulation Time=1.368 ms Event Id=162ARP---ARP_ReplySent
EventTime:1368.340 SrcIP:192.168.2.5 MAC_Add:AAAAAAAAAAAB DestIP:1
92.168.2.4
0 % is completed... Simulation Time=1.458 ms Event Id=171ARP---ARP_ReplyGot
EventTime:1457.633 SrcIP:192.168.2.4 DestIP:192.168.2.5 MAC_Add:
AAAAAAAAAAAB
1 % is completed... Simulation Time=119.938 ms Event Id=3097
Breakpoint triggered...
Press any key to continue the process...

Press enter then control goes to the project and stops at the break point in the source code as shown below.



All debugging options like **step over (F10), step into (F11), step out (Shift + F11), continue(F5)** are available.

After execution of the function, the control goes back to NetSim and then comes back to the custom code the next time the function is called in the simulation.

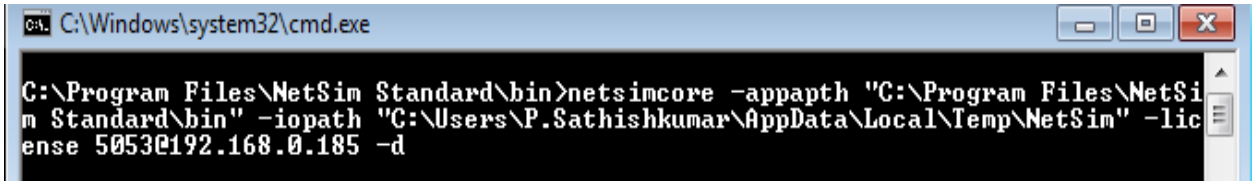
To stop debugging and continue execution, remove all breakpoint and press F5 (key). This thengives the control back to NetSim, for normal execution to continue.

Event Trace - Writingfile name and line number of the code which generated the event

To debug your own (custom) code, it is often helpful to know which section of the code (file name & line number) generated the event under study. To use this feature -

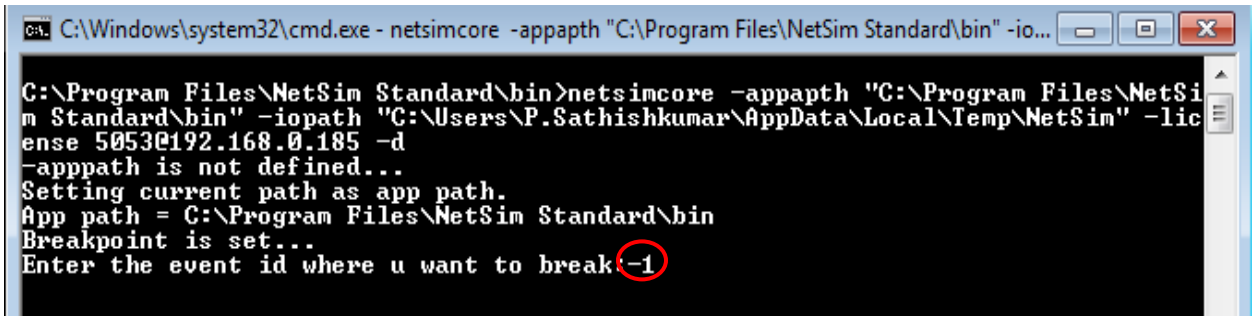
Step 1: Enable the event trace.

Step 2: Run the NetSim via CLI in debug mode (Refer NetSim Help ->Running Netsim via CLI) with –d as the fourth parameters



Press enter

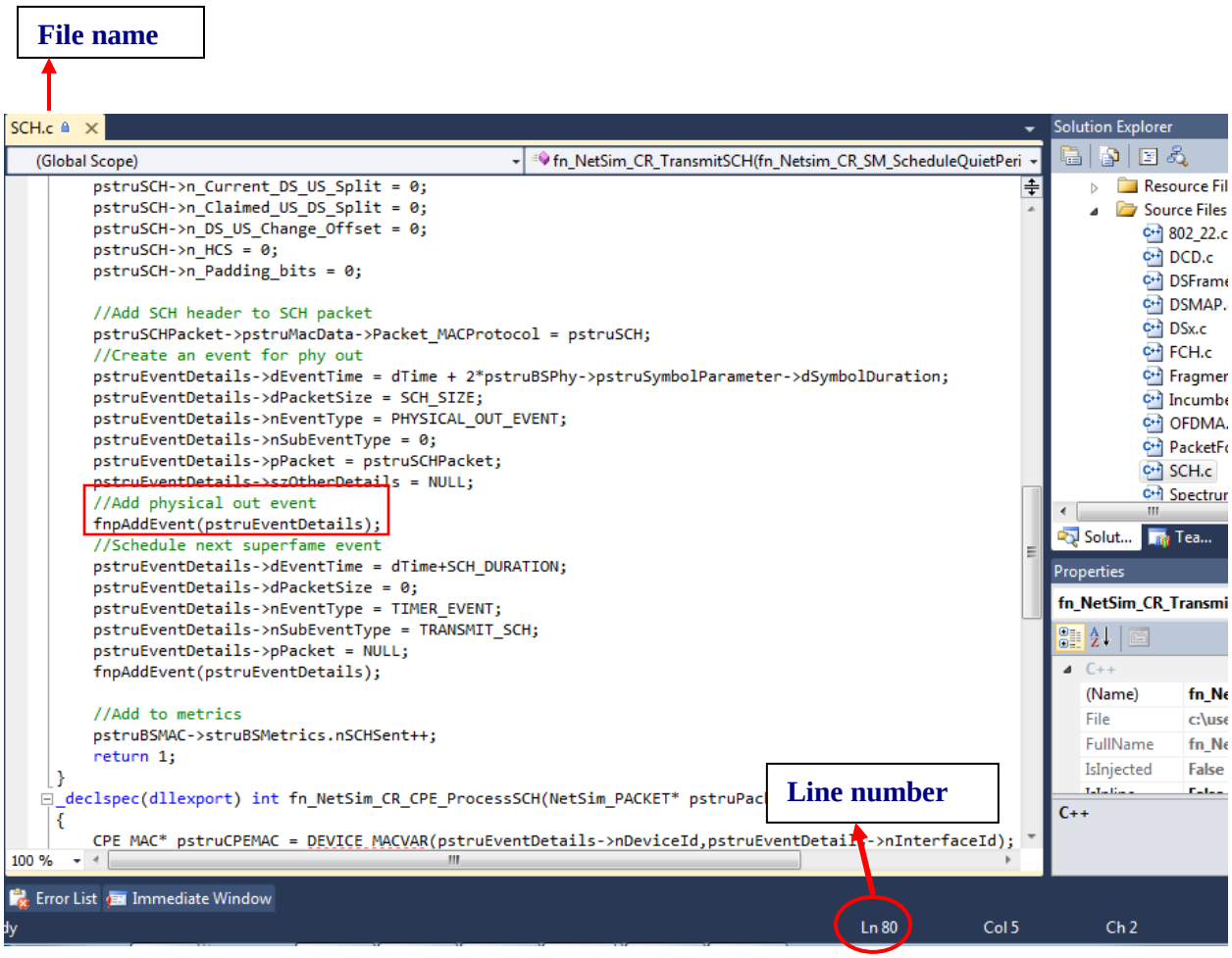
Step 3: Enter -1 as the event ID


C:\Windows\system32\cmd.exe - netsimcore -appath "C:\Program Files\NetSim Standard\bin" -io...
C:\Program Files\NetSim Standard\bin>netsimcore -appath "C:\Program Files\NetSi
m Standard\bin" -iopath "C:\Users\P.Sathishkumar\AppData\Local\Temp\NetSim" -lic
ense 5053@192.168.0.185 -d
-apppath is not defined...
Setting current path as app path.
App path = C:\Program Files\NetSim Standard\bin
Breakpoint is set...
Enter the event id where u want to break-1

Upon running, NetSim will write the file name and line number of the source code that generated each event.

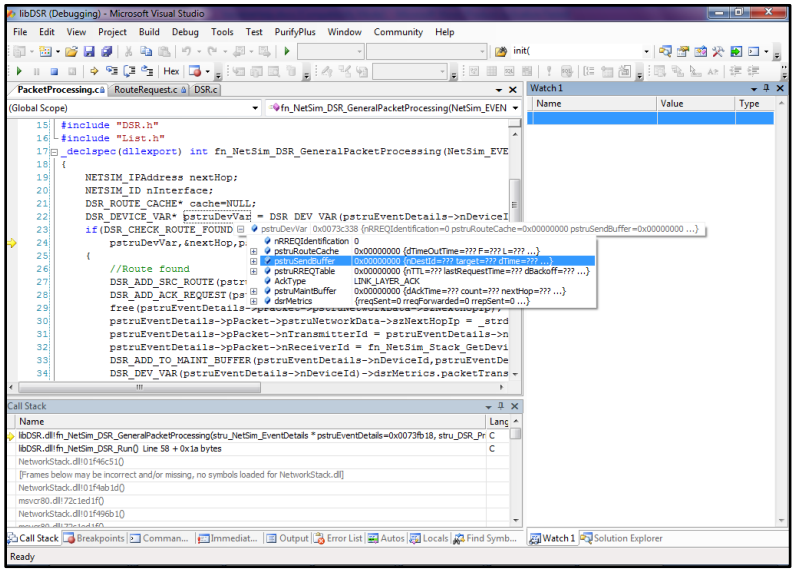
A	B	C	D	E	F	G	H	I	J	K	L	M	N
Event_Id	Event_Type	Device_Id	Packet_Id	Subevent_Type	Packet_Size	Prev_Eve	Line_No	File_Name					
2	TIMER_EVENT	1	0	TRANSMIT_SCH	0		1888	.\802_22_lib.c					
4	PHYSICAL_OUT	1	SCH	0	45		80	c:\users\p.sathishkumar\desktop\libcognitiveradio\sch.c					
6	PHYSICAL_IN	2	SCH	0	45		2082	.\802_22_lib.c					
7	PHYSICAL_IN	3	SCH	0	45		2082	.\802_22_lib.c					
8	TIMER_EVENT	1	0	FORM_DS_BURS	0		212	c:\users\p.sathishkumar\desktop\libcognitiveradio\802_22.					
9	MAC_IN	2	SCH	0	45		303	c:\users\p.sathishkumar\desktop\libcognitiveradio\802_22.					
10	MAC_IN	3	SCH	0	45		303	c:\users\p.sathishkumar\desktop\libcognitiveradio\802_22.					
11	TIMER_EVENT	1	0	TRANSMIT_DS_B	0		43	c:\users\p.sathishkumar\desktop\libcognitiveradio\fch.c					
13	PHYSICAL_OUT	1	FCH	0	3	1	118	c:\users\p.sathishkumar\desktop\libcognitiveradio\dsfram					
14	PHYSICAL_IN	2	FCH	0	3	1	2082	.\802_22_lib.c					
15	PHYSICAL_IN	3	FCH	0	3	1	2082	.\802_22_lib.c					
16	TIMER_EVENT	1	0	TRANSMIT_DS_B	0	1	246	c:\users\p.sathishkumar\desktop\libcognitiveradio\802_22.					

Note: In the above trace file Event Id 2 is triggered inside the 802_22_lib.c file which is present in Cognitive radio.lib.Since all the lib files are opaque to the end user, you cannot see the source code of the lib file. However, Event Id 4 is triggered at line number 80 of sch.c file and you can find the location of the event by opening the sch.c file as shown below.



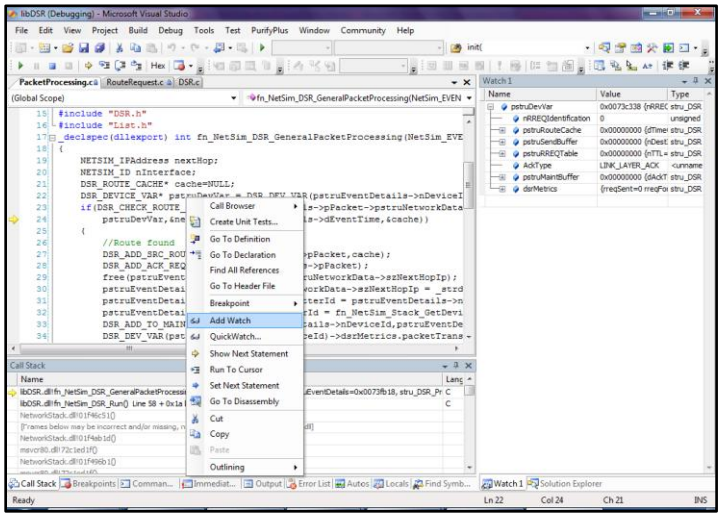
Viewing variables while debugging code -

To see the value of a variable, when debugging hover the mouse over the variable name in the code. A text box with variable contents appears. If the variable is a structure and contains other variables, then click on the plus sign which is there to the left of the text box. Users can pin the variable to watch by clicking on the pin icon to the right of that variable in the text box.



Adding the variable to watch -

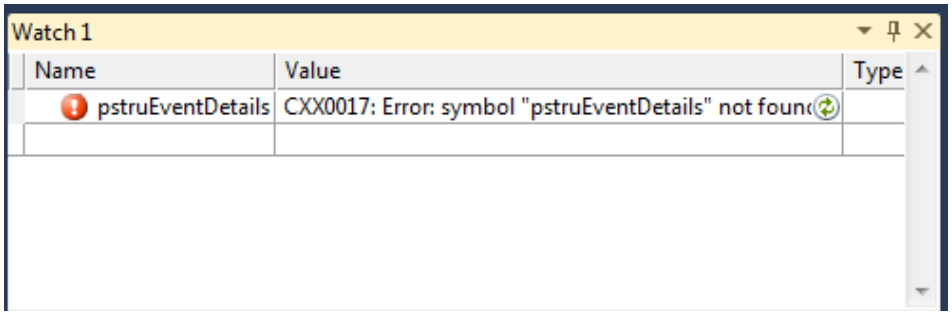
Watch the change in the variable as the code progress by right clicking on the variable & clicking on "add watch" tab. This is useful if to continuously monitor the change in the variable as the code progresses.



Viewing external variables -

During the process of debug users would come acrossvariables that are defined outside the source file being built as a .dll. Such variables cannot be viewed directly when added in the watch tab, as this would throw the error

CX0017: Error:symbol “Variable_Name”not found.

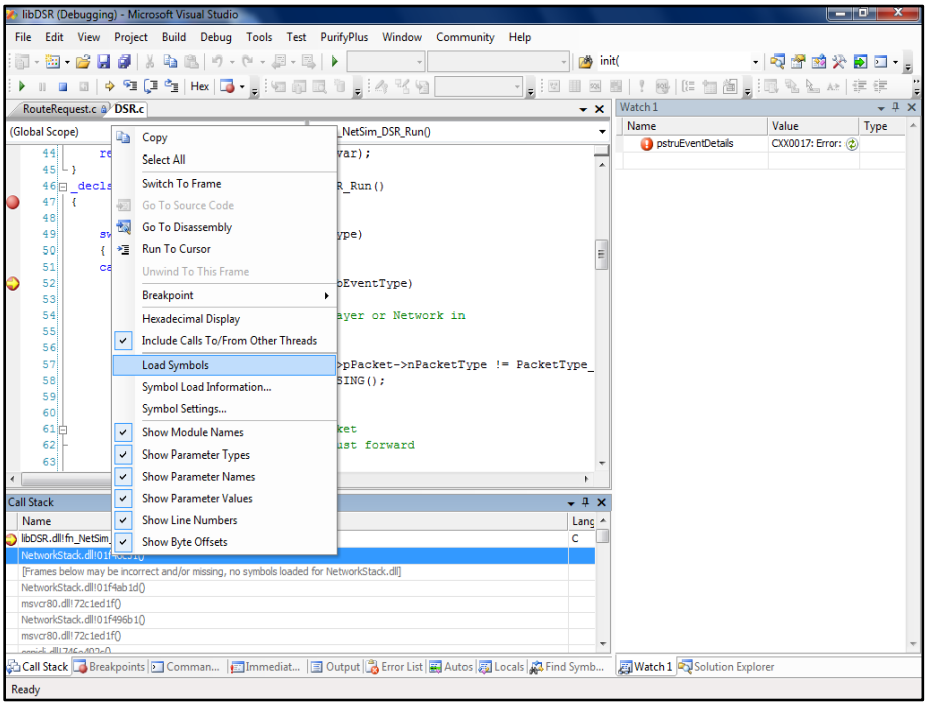


In the call stack window one can find the file in which that variable is situated. Right click on the dll file name in the call stack window, in this case NetworkStack.dll. Then in the pull down menu which appears, select "load symbols from" and give the path of the pdb(program database) file.

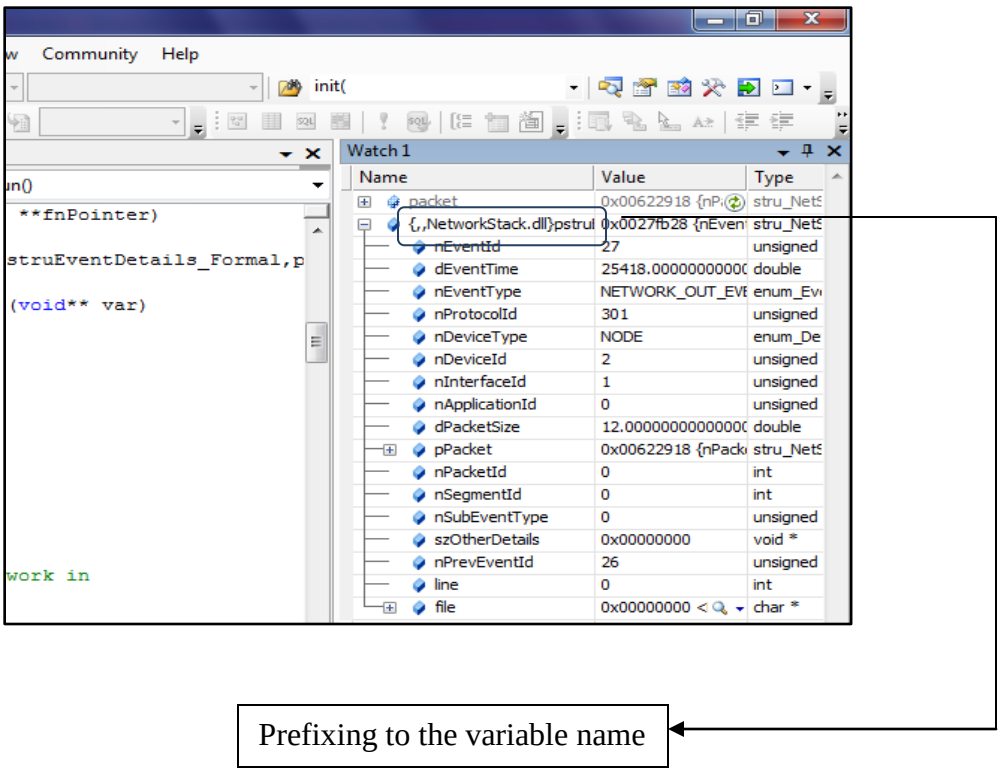
A program database (.pdb) file, also called a symbol file, maps the identifiers that a user creates in source files for classes, methods, and other code to the identifiers that are used in the compiled executablesof the project. The .pdb file also maps the statements in the source code to the execution instructions in the executables. The debugger uses this information to determine: the source file and the line number displayed in the Visual Studio IDE and the location in the executable to stop at when a user sets a breakpoint. A symbol file also contains the original location of the source files, and optionally, the location of a source server where the source files can be retrieved from.

When a user debugs a project in the Visual Studio IDE, the debugger knows exactly where to find the .pdb and source files for the code. If the user wants to debug code outside their project source code, such as the Windows or third-party code the project calls, the user has to specify the location of the .pdb (and optionally, the source files of the external code) and those files need to exactly match the build of the executables.

The pdb files are usually available in NetSim's install directory, else write to support@tetcos.com for the latest copy of these debug files.



In the watch window, the variable which the user has to watch should be edited by double clicking on it and prefixing {,,NetworkStack.dll} to the variable name and pressing enter. (The name of the respective file in which the variable is defined should be mentioned - in this case NetworkStack.dll).



Accessing External Variables –

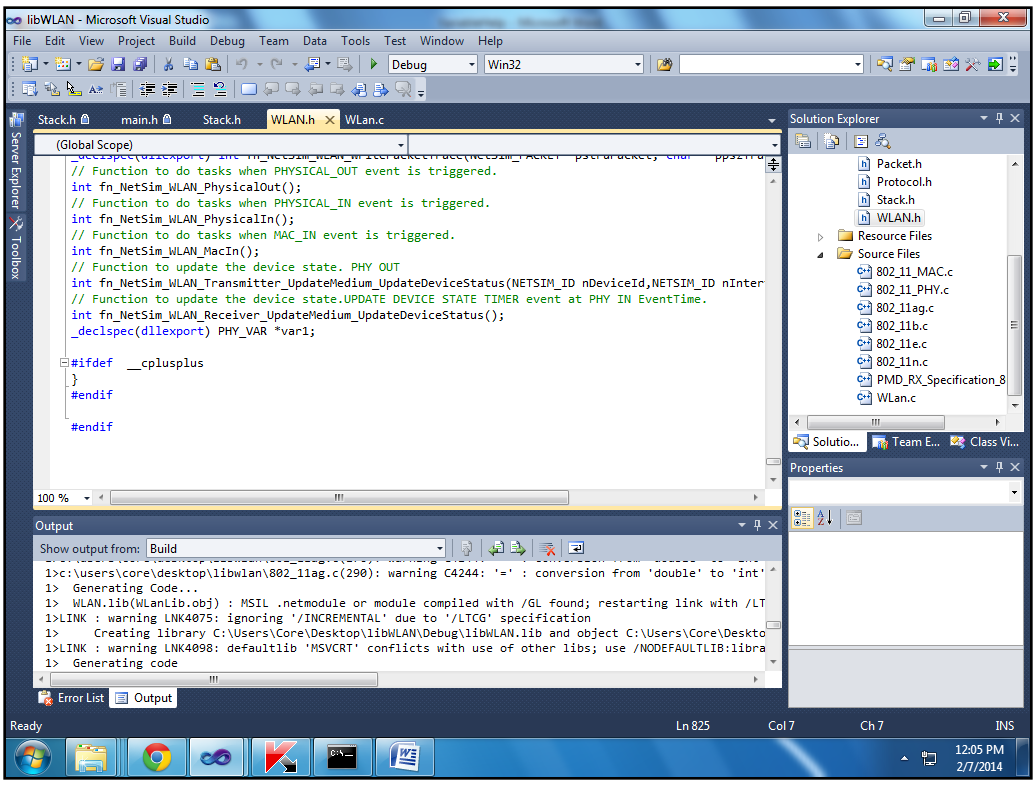
In NetSim, while a scenario is simulated, it is possible to access variables which are defined in one .dll file from another .dll file. An example is given below showing how a WLAN file variable dTotalReceivedPower can be accessed in the DSR file.

The user must create a WLAN solution in visual studio (as mentioned above in how to write custom codes). In the example below, the libWLAN folder path is D:\libWLAN.

The variable dTotalReceivedPower is defined in a structure PHY_VAR. So the user will have to access a pointer of type PHY_VAR. In the header file where the structure definition is given, the following line of code must be written –

_declspec(dllexport) PHY_VAR *var1;

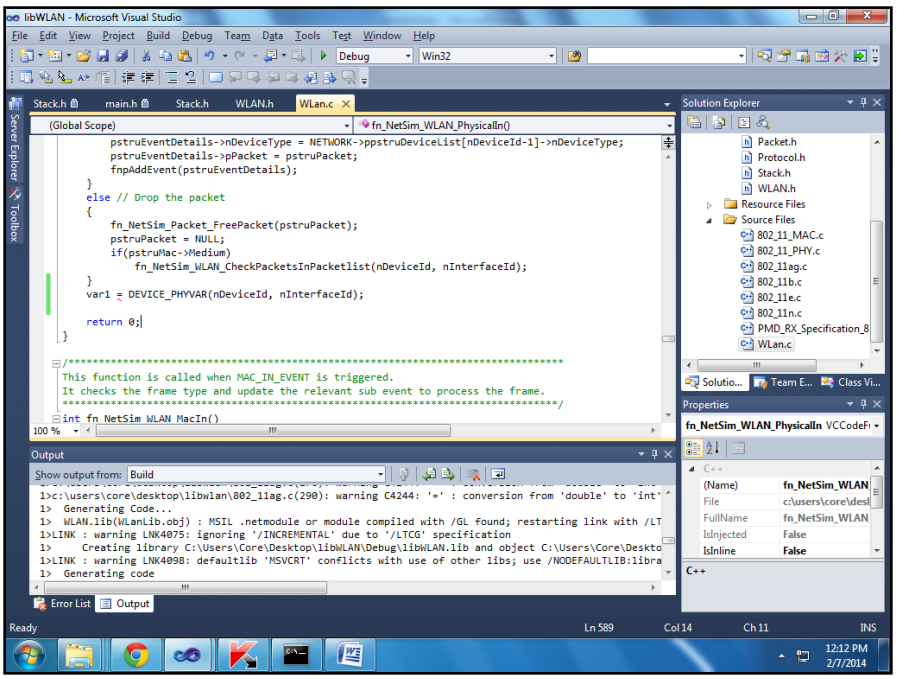
In the example, the code line must be written in WLAN.h file.



In the main function where a user wishes to find the dTotalReceivedPower, the variable must be assigned the respective value. In the above case, the following line of code must be written in WLAN.c file

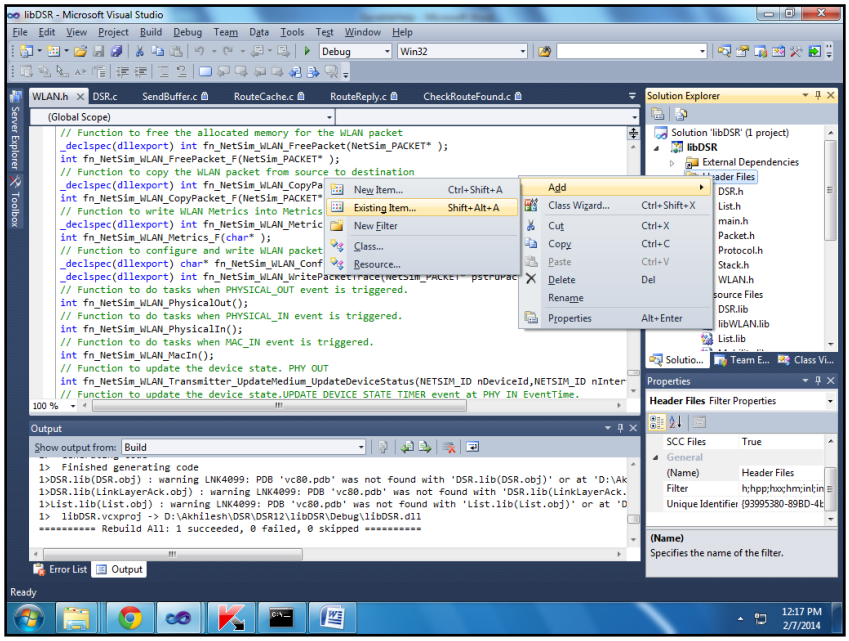
var1 = DEVICE_PHYVAR(nDeviceId, nInterfaceId);

Note that the parameters given in the macro or any function which assigns a value to the variable must be defined beforehand in the code. Here nDeviceId and nInterfaceId are defined beforehand. The solution must be built and the resulting D:\libWLAN\Debug\libWLAN.dll file which gets created must be copied in to the NetSim standard bin folder (Again refer to writing custom code in NetSim).

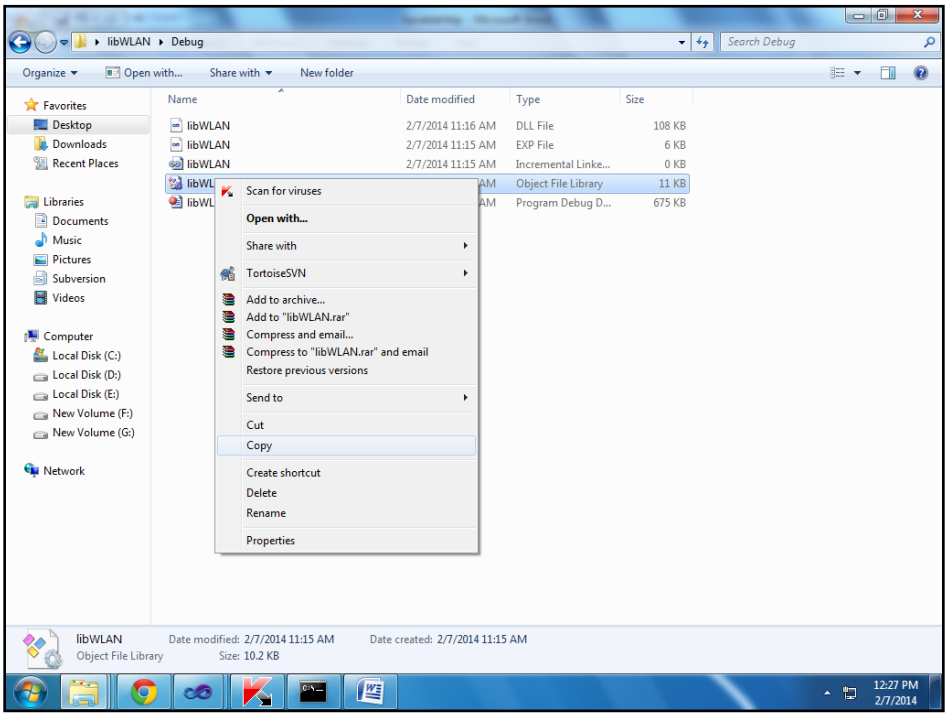


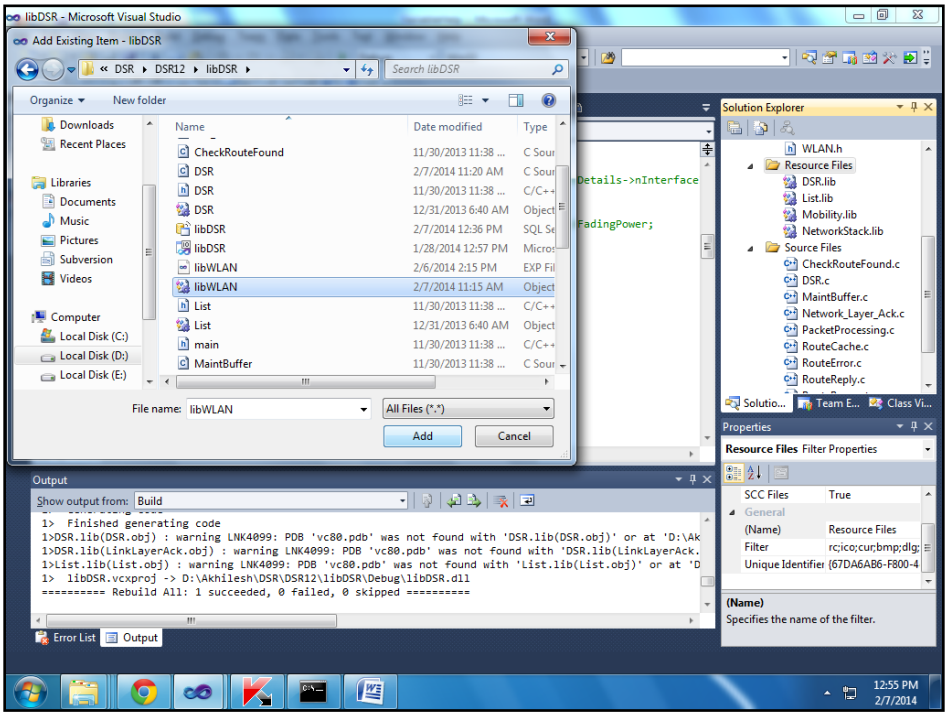
The user must create a DSR solution as mentioned above (How to write a custom code.) In the example, the folder path is D:\libDSR.

The D:\libWLAN\WLAN.h file which was changed earlier must be copied and pasted in the DSR solution folder D:\libDSR and must be included in the header files in DSR solution in visual studio.



The library file D:\libWLAN\Debug\libWLAN.lib file which got created must be copied and pasted in the DSR solution folder D:\libDSR and must be included as a resource file in DSR solution in visual studio.

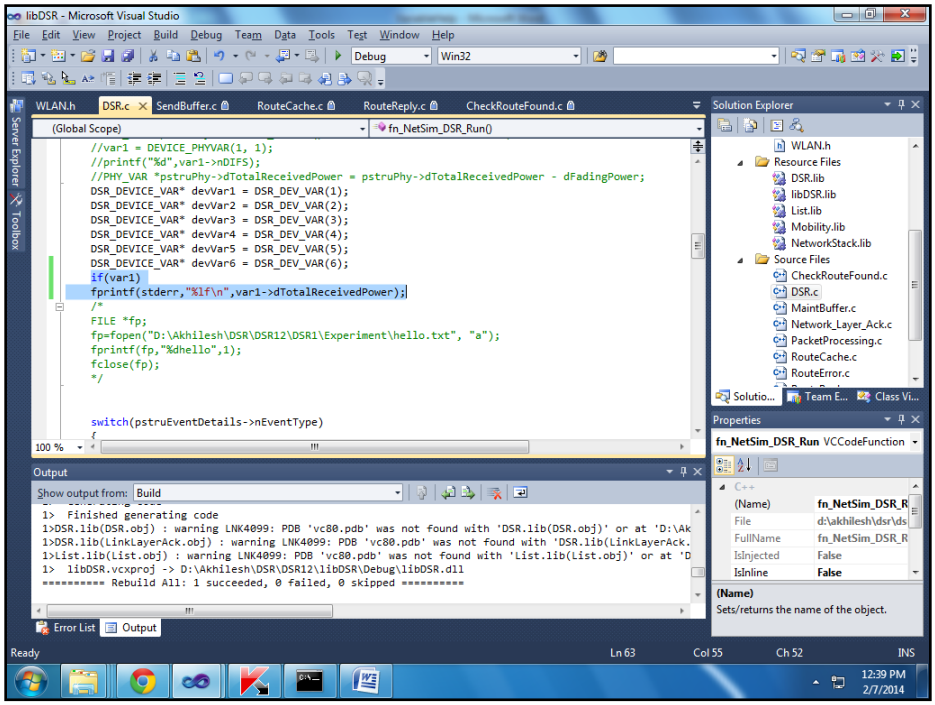




The WLAN.h header file which was included in DSR solution must be edited and the line where _declspec(dllexport) PHY_VAR *var1; was written must be replaced with _declspec(dllimport) PHY_VAR *var1;

For viewing the variable value in the command prompt, the following lines must be added in DSR.c.

```
#include “WLAN.h”
if(var1)
fprintf(stderr,"%lf\n",var1->dTotalReceivedPower);
```



The solution must be built and the resulting dll file D:\libDSR\Debug\libDSR.dll must be copied and replaced in the NetSim Standard bin path. When a scenario is run, the Total Received Power can be seen in the command prompt.

NetSim APIs

NetSim provides a wide variety of APIs for protocol developers. These are available in

1. packet.h – Packet related APIs (Eg: Create_Packet ( ), Copy_Packet ( ), Free_Packet ( ) etc.)
2. stack.h – Network / device / link and event related APIs (Eg: Get_Device_IP ( ), Get_Connected_Link( ), Add_Event ( ) etc.)
3. list.h, -- Optimized list operation calls since NetSim uses lists extensively (Eg: Add_to_list ( ), Sort_list ( ) etc.)
4. NetSim_Graph.h – This is used for plotting graphs using GNU plot
5. IP_Addressing.h – For setting & getting IP address per the appropriate format (Eg: Convert_IP_to_string ( ), Compare_IP( ) etc.)

For detailed help please refer the appropriateheader (.h) files inside:
../NetSim_Standard/src/simulation/include or read through the doxygen source code documentation available inside NetSim -> Help -> NetSim source code Help

- Include all the header (.h) files from the include folder
- NetworkStack.lib is a "import library" file and has the definitions for the functions present in the NetworkStack.dll
- When developing new protocols users should create their own protocol.h and declare all the protocol specific variables here. Stack & packet related variables should be used from stack.h and packet.h

NetSim Network Stack calling individual Protocol

Every protocol should provide the following APIs as hooks to the network stack:

- int (*fn_NetSim_protocol_init)(conststruct stru_NetSim_Network*,conststruct stru_NetSim_EventDetails*,constchar*,constchar*,int,constvoid**);
  Using this API the stack passes all the relevant pointers to variables, paths etc needed for the protocol. Inside this function a) local variables should be initialized, b) Initial events if any should be written,eg: Hello packet in RIP, STP in Ethernet c) File pointers for reading & writing protocol_specific_IO files.

- int (*fn_NetSim_protocol_Configure)(conststruct stru_NetSim_Network*, int nDeviceId, int nINterfaceID, int nlayertype, fnpAllocateMemory, fnpFreeMemory, fpConfigLog);
  The stack calls this API when reading the config file. Upon reaching the appropriate protocol definition in the XML file, the stack calls this and passes all these pointers to the protocol

- int (*fn_NetSim_protocol_run)(): This is called by the stack to run the protocol

- char* (*fn_NetSim_protocol_trace)(int): This called by the stack to write the event trace

- int (*fn_NetSim_protocol_CopyPacket)(const NetSim_PACKET* pstruDestPacket,const NetSim_PACKET* pstruSrcPacket):
  This is for copying protocol specific parameters / data into the packed

- int (*fn_NetSim_protocol_FreePacket)(const NetSim_PACKET* pstruPacket): The this to free the protocol specific parameters / data in the packet

- (*fn_NetSim_protocol_Metrics)(const FILE* fpMetrics): This is to write the metrics file upon completion of the simulation

- int (*fn_NetSim_protocol_Finish)(): To release all memory after completion

- char* (*fn_NetSim_protocol_ConfigPacketTrace)(constvoid* xmlNetSimNode); To configure the packet packet trace in terms of the parameters to be logged

- char* (*fn_NetSim_protocol_WritePacketTrace)(const NetSim_PACKET*); To configure the event trace in terms of the parameters to be logged

Advanced Features

Random Number Generator and Seed Values

All network simulations involve an element of randomness. Some examples are -

a. It is possible to configure the traffic sources in the simulation to generate traffic in a perfectly regular pattern. However, this is typically not the case in the real world.
b. Node back-off's after collisions are random to resolve contention issues
c. The exact bit which is errored, based on Bit error probability of a wireless channel, is decided randomly

NetSim uses an in-built Linear Congruential Random Number Generator (RNG) to generate the randomness. The RNG uses two seeds values to initialize the RNG.

Having the same set of seed values ensures that for a particular network configuration the same output results will be got, irrespective of the PC or the time at which the simulation is run. This ensures repeatability of experimentation.

Modifying the seed value will lead to the generation of a different set of random numbers and thereby lead to a different sequence of events in NetSim. When simulations are run for a network configuration with different seed values, the results will likely be slightly different.

More advanced users may note that "Confidence" can be established by analyzing a set of results with different seed values for the same network scenario.

How to Fail a Node in NetSim

In this section we explain node failure using MANET-DSR as an example

- Using Device Id: Identify the Device ID of the particular device.
- The fn_NetSim_DSR_Run( ) is the main function to handle all the protocol functionalities

DSR_Run( )

```
strcpy(pszFilepath,pszIOPath);
        strcat(pszFilepath,"/Output.txt");
        fp = fopen(pszFilepath,"r");
        i=0;
        if(fp)
        {
                while(fgets(pszConfigInput,500,fp)!= NULL)
                {
                        sscanf(pszConfigInput,"%d    %d",&NodeArray[i],&TimeArray[i]);
                        i+=1;
                }
        }

for(i=0;i<100;i++)
        if((pstruEventDetails->nDeviceId == NodeArray[i])&&(pstruEventDetails->dEventTime
>= TimeArray[i]))  // Node failure happens after a value of time storted in an array
        {
        // This just returns instead of running DSR functionality when the node is failed
                pstruEventDetails->nInterfaceId = 0;
                return 0;
        }
```

Note: Add the above lines of code after the variable declarations in the function

Build the .dll by clickingBuild→Build Solution.(The dll will be created inside the Visual studio project →Debug)

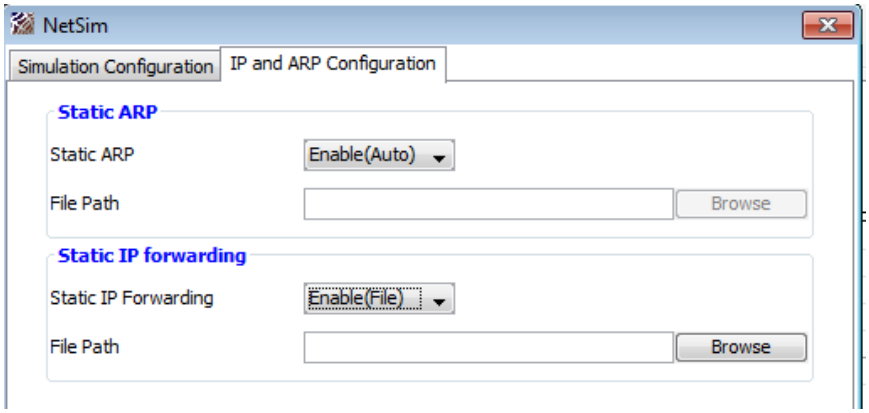
Static Routing (IP Forwarding)

How to Setup Static Routes in RIP and OSPF

Using GUI

Step1: Create a scenario in internetworks and set properties of the devices. Configure the applications.

Step 2: While running the simulation, go to IP and ARP configuration tab and enable Static IP forwarding and browse the path where it has to be saved.



Using CLI

Step1: In internetworks, Static Routes can be set for any scenario having a minimum of 3 routers, 2 switches and 3 wired nodes. The easiest way to do this is to first run the scenario with routing protocol set as RIP/OSPF and save the Configuration file

Step 2: Open the Configuration file with Visual studio. Expand the Router configuration and set the Static Routing information in Application Layer property of the device, by enabling the Static routing status. Then set the appropriate Static routing file name and file path.

RIP

```
By default:
<LAYERTYPE="APPLICATION_LAYER">
<ROUTING_PROTOCOLNAME="RIP"SETPROPERTY="true">
<PROTOCOL_PROPERTYFILE_NAME="StaticRouting.txt"FILE_PATH="C:\Program
Files\NetSim
```

```
standard\Docs\Sample_Configuration\Internetworks"GARBAGE_COLLECTION_TIMER="12
0"STATIC_ROUTING_STATUS="DISABLE"TIMEOUT_TIMER="180"UPDATE_TIMER="30"VERSION=
"2"/>
</ROUTING_PROTOCOL>
</LAYER>
 Change to:
<LAYERTYPE="APPLICATION_LAYER">
<ROUTING_PROTOCOLNAME="RIP"SETPROPERTY="true">
<PROTOCOL_PROPERTYFILE_NAME="StaticRouting.txt"FILE_PATH="C:\Program
Files\NetSim
standard\Docs\Sample_Configuration\Internetworks"GARBAGE_COLLECTION_TIMER="12
0"STATIC_ROUTING_STATUS="ENABLE"TIMEOUT_TIMER="180"UPDATE_TIMER="30"VERSION="
2"/>
</ROUTING_PROTOCOL>
</LAYER>
```

OSPF

```
By default:
<LAYERTYPE="APPLICATION_LAYER">
<ROUTING_PROTOCOLNAME="OSPF"SETPROPERTY="true">
<PROTOCOL_PROPERTYFILE_NAME="StaticRouting.txt"FILE_PATH="C:\Program
Files\NetSim standard\Docs\Sample_Configuration\Internetworks"
LS_REFRESH_TIME="1800"MAX_AGE="3600"ROUTER_PRIORITY="1"STATIC_ROUTING_STATUS=
"DISABLE"VERSION="2"/>
</ROUTING_PROTOCOL>
</LAYER>
Change to:
<LAYERTYPE="APPLICATION_LAYER">
<ROUTING_PROTOCOLNAME="OSPF"SETPROPERTY="true">
<PROTOCOL_PROPERTYFILE_NAME="StaticRouting.txt"FILE_PATH="C:\Program
Files\NetSim standard\Docs\Sample_Configuration\Internetworks"
LS_REFRESH_TIME="1800"MAX_AGE="3600"ROUTER_PRIORITY="1"STATIC_ROUTING_STATUS=
"ENABLE"VERSION="2"/>
</ROUTING_PROTOCOL>
</LAYER>
```

Note:

1. Update this information in any one of the router
2. A sample StaticRouting.txt file will be available inside “C:\Program Files\NetSim standard\Docs\Sample_Configuration\Internetworks”. Appropriately modify it for the scenario.

The StaticRouting.txt file contains

1. Device Id of the Router
2. List of entries to add in the routing table

DEVICE_ID:1

ip route 192.168.0.0 255.255.255.0 192.168.0.1 [1]

ip route 192.168.1.0 255.255.255.0 192.168.1.1 [1]

DEVICE_ID:2

ip route 192.168.2.0 255.255.255.0 192.168.2.1 [1]

ip route 192.168.3.0 255.255.255.0 192.168.3.1 [1]

The above format is perstandard

Cisco command for static routes

iproute *dest_ip maskgateway_ip* [*distance*]

1. The **ip route** is the Cisco command to add the static route
2. The *dest_ip* is the Network address for the destination network
3. The **mask** is the Subnet mask for the destination network
4. The **gateway_ip** is the IP address of the next-hop router
5. The **distance**is the administrative distance for the route. The default is 1.

Step 3: After this, run this configuration file through CLI and static routes will be used for routing.

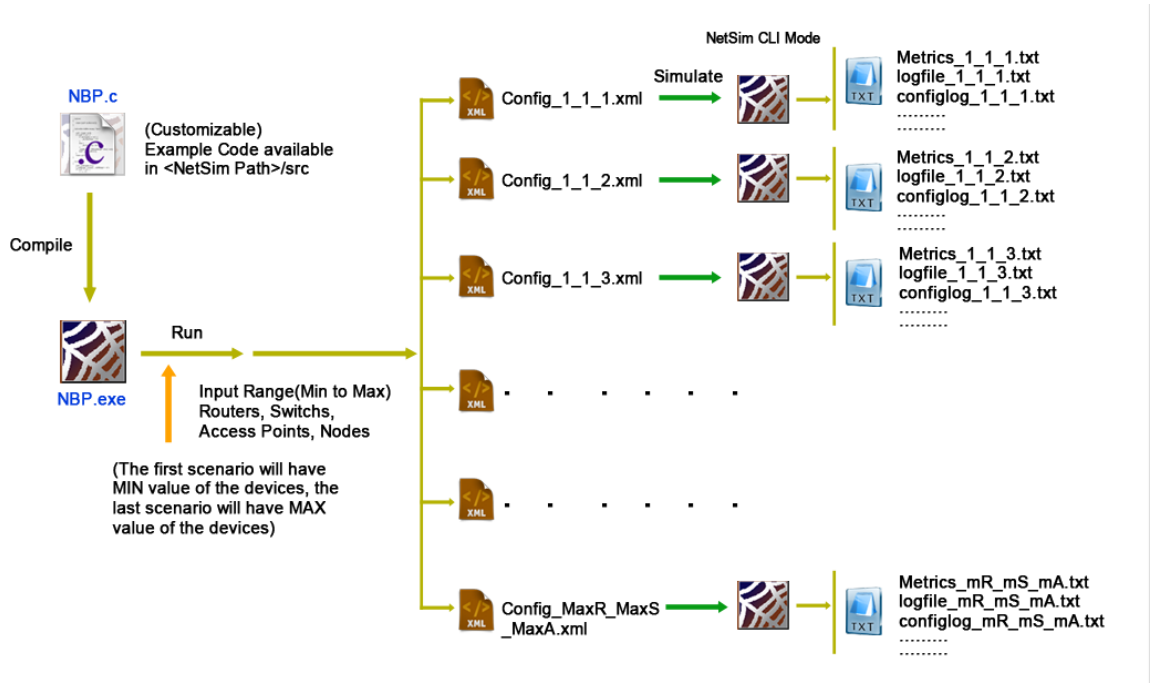
NetSim – Batch Processing

Batch processing is the execution of series of simulations without manual intervention.

Consider the example, where a user wishes to create and simulate thousands of network scenarios and store & analyze the performance metrics of each simulation run. It is not possible to do this using the GUI. However, the same can be done when writing a batch processing program which runs NetSim via CLI.

An example C program for batch processing of NetSim can be found at <NetSim Path>\src\NetSim_Batch_Processing.c where <NetSim Path> is the path to where NetSim is installed. This batch program can be customized by users as desired.

Batch File Picture



Given below are important sections of the source code of this batch program.

```
#define apppath "C:\\Program Files (x86)\\NetSim pro7\\bin"    // change to your app path
#define iopath "F:\\NetSim_Batch_Processing"                   // change to your IO Path
#define license "5053@192.168.0.10"          // change to where your license server is running
```

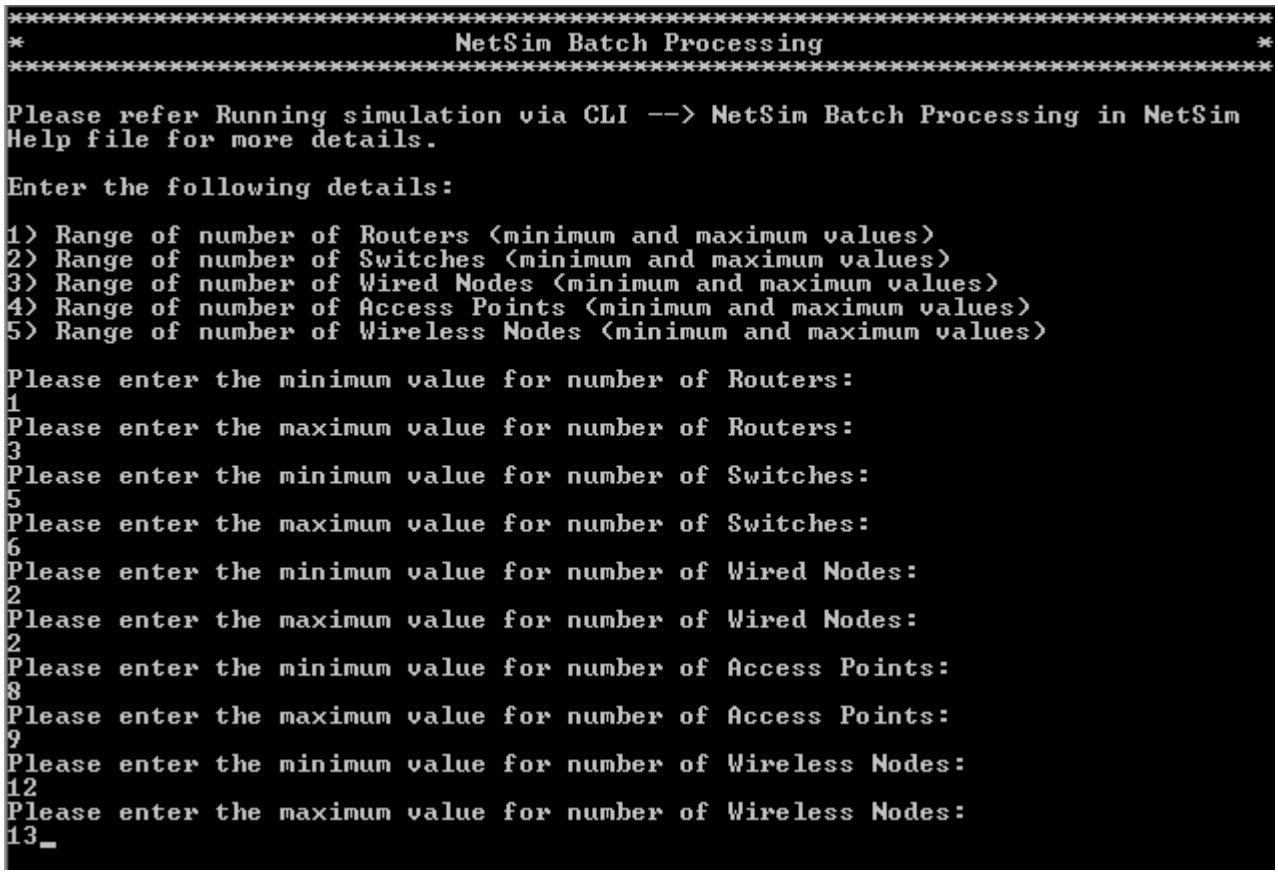
Flow of the batch program code and important functions

```
main ( )
{
        {// iterate from MIN to MAX for all of the devices to create configuration files
        If      { /* If certain network validation conditions are not met write the name of
                configuration file to NetSim_Batch_Log.txtNetSim_Bact_Log.txt contains all the
                config files that did not meet the  neccassary validation conditions and were not
                created duing the batch execution */
                }
        Else    {
                        //Create configuration file

                        fnCreateExperimentInformation(fpConfigurationFilePointer);
                        fnCreateNetworkConfiguration(fpConfigurationFilePointer);
                        fnCreateSimulationParameter(fpConfigurationFilePointer);
                        fnCreateStatisticsCollection(fpConfigurationFilePointer);

                }
        } // End of creation of all config files
        Create batch file RunAll.bat

        Run the batch file RunAll.bat // Loop over number of configuration files, call NetSim
                                      // executable which will write the output files
} // Main end
fnCreateNetworkConfiguration(FILE* fpConfigurationFilePointer){

        fnCreateDeviceConfiguration(fpConfigurationFilePointer);
        fnCreateConnectionConfiguration(fpConfigurationFilePointer);

}
```

```
fnCreateDeviceConfiguration(FILE* fpConfigurationFilePointer){

        fnConfigureRouters(fpConfigurationFilePointer);
        fnConfigureSwitches(fpConfigurationFilePointer);
        fnConfigureWiredNodes(fpConfigurationFilePointer);
        fnConfigureAccessPoints(fpConfigurationFilePointer);
        fnConfigureWirelessNodes(fpConfigurationFilePointer);
}

fnCreateConnectionConfiguration(FILE* fpConfigurationFilePointer){

        fnCreateRouterRouterLinks(fpConfigurationFilePointer);
        fnCreateRouterSwitchLinks(fpConfigurationFilePointer);
        fnCreateSwitchWiredNodeLinks(fpConfigurationFilePointer);
        fnCreateSwitchAccessPointLinks(fpConfigurationFilePointer);
        fnCreateAccessPointWirelessNodesLinks(fpConfigurationFilePointer);
}

fnConfigureRouters(FILE* fpConfigurationFilePointer){

        {  // iterate until all routers are configured

                fnCreateRouterInterfaces(fpConfigurationFilePointer, nLoopCounterI);
                fnCreateRouterTransportLayer(fpConfigurationFilePointer);
                fnCreateRouterApplicationLayer(fpConfigurationFilePointer);
        }
}

fnConfigureSwitches(FILE* fpConfigurationFilePointer){

        {  // iterate until all switches are configured

                fnCreateSwitchInterfaces(fpConfigurationFilePointer, nLoopCounterI);

        }
}

fnConfigureWiredNodes(FILE* fpConfigurationFilePointer){

        {  // iterate until all wired nodes are configured

                fnCreateWiredNodeInterface(fpConfigurationFilePointer, nLoopCounterI);
                fnCreateWiredNodeTransportLayer(fpConfigurationFilePointer);
                fnCreateWiredNodeApplicationLayer(fpConfigurationFilePointer,
                nLoopCounterI);

        }
}
```

```
fnConfigureAccessPoints(FILE* fpConfigurationFilePointer){

        {  // iterate until all access points are configured

                fnCreateAccessPointInterfaces(fpConfigurationFilePointer, nLoopCounterI);

        }
}
fnConfigureWirelessNodes(FILE* fpConfigurationFilePointer){

        {  // iterate until all wireless nodes are configured

                fnCreateWirelessNodeInterface(fpConfigurationFilePointer, nLoopCounterI);
                fnCreateWirelessNodeTransportLayer(fpConfigurationFilePointer);
                fnCreateWirelessNodeApplicationLayer(fpConfigurationFilePointer,
                nLoopCounterI);
        }
}
```

On compiling NetSim_Batch_Processing.c and running the executable, the following screen will appear:


NetSim Batch Processing
Please refer Running simulation via CLI --> NetSim Batch Processing in NetSim Help file for more details.
Enter the following details:
1) Range of number of Routers (minimum and maximum values)
2) Range of number of Switches (minimum and maximum values)
3) Range of number of Wired Nodes (minimum and maximum values)
4) Range of number of Access Points (minimum and maximum values)
5) Range of number of Wireless Nodes (minimum and maximum values)
Please enter the minimum value for number of Routers:
1
Please enter the maximum value for number of Routers:
3
Please enter the minimum value for number of Switches:
5
Please enter the maximum value for number of Switches:
6
Please enter the minimum value for number of Wired Nodes:
2
Please enter the maximum value for number of Wired Nodes:
2
Please enter the minimum value for number of Access Points:
8
Please enter the maximum value for number of Access Points:
9
Please enter the minimum value for number of Wireless Nodes:
12
Please enter the maximum value for number of Wireless Nodes:
13

Sample Input:

Device	Minimum	Maximum
Routers	1	3
Switches	5	6
Wired Nodes	2	2
Access Points	8	9
Wireless Nodes	12	13

On entering the above details the following configuration files are created and simulation is run for all these configuration files.

Routers	Switches	Wired Nodes	Access Points	Wireless Nodes	Configuration file
1	5	2	8	12	Configuration1.5.2.8.12.xml
1	5	2	8	13	Configuration1.5.2.8.13.xml
1	5	2	9	12	Configuration1.5.2.9.12.xml
1	5	2	9	13	Configuration1.5.2.9.13.xml
1	6	2	8	12	Configuration1.6.2.8.12.xml
1	6	2	8	13	Configuration1.6.2.8.13.xml
1	6	2	9	12	Configuration1.6.2.9.12.xml
1	6	2	9	13	Configuration1.6.2.9.13.xml
2	5	2	8	12	Configuration2.5.2.8.12.xml
2	5	2	8	13	Configuration2.5.2.8.13.xml
2	5	2	9	12	Configuration2.5.2.9.12.xml
2	5	2	9	13	Configuration2.5.2.9.13.xml
2	6	2	8	12	Configuration2.6.2.8.12.xml
2	6	2	8	13	Configuration2.6.2.8.13.xml
2	6	2	9	12	Configuration2.6.2.9.12.xml
2	6	2	9	13	Configuration2.6.2.9.13.xml
3	5	2	8	12	Configuration3.5.2.8.12.xml
3	5	2	8	13	Configuration3.5.2.8.13.xml
3	5	2	9	12	Configuration3.5.2.9.12.xml
3	5	2	9	13	Configuration3.5.2.9.13.xml
3	6	2	8	12	Configuration3.6.2.8.12.xml
3	6	2	8	13	Configuration3.6.2.8.13.xml
3	6	2	9	12	Configuration3.6.2.9.12.xml
3	6	2	9	13	Configuration3.6.2.9.13.xml

Output Files:

Configuration1.5.2.8.12.xml_Metrics.txt is the metrics file created for configuration file Configuration1.5.2.8.12.xml. Similarly other output files like Configuration1.5.2.8.12.xml_log_file.txt are created. Output files for all the configuration files with the name of the configuration file as the prefix are generated.

All the configuration files which are not created due to various reasons (explained in the programs code comments) are listed in NetSim_Batch_Log.txt.

SINR, BER and Propagation models for 802.15.4

SINR Calculation:

Analogous to the SNR used often in wired communications systems, the SINR is defined as the power of a certain signal of interest divided by the sum of the interferencepower (from all the other interfering signals) and the power of some background noise.
The interference power is the difference between the total power received by the receiver and the power received from one particular transmitter.

The background thermal noise in dBm at room temperature is given by:

$$\text{P (in dBm)} = -174 + 10 \times \log_{10}(\Delta f)$$

where Δf is the Bandwidth in Hertz. For 802.15.4, Δf = 2 MHz

$$\text{P (in mW)} = 10^{\left(\frac{P\ (in dBm\ )}{10}\right)}$$

Therefore, SINR in dBm is calculated as:

$$\text{SINR (in dBm)} = \log_{10}\left(\frac{\text{Received power (in mW)}}{\text{Interference Noise (in mW)} + \text{Thermal Noise (in mW)}}\right)$$

Bit Error Rate (BER) Calculation:

The bit error rate (BER) is the number of bit errors divided by the total number of transferred bits during a studied time interval. The BER results were obtained using the analytical model from IEEE standard 802.15.2-2003 [B9]. The calculation follows the approach outlined in 5.3.2 of that standard.

$$\text{BER} = \left(\frac{8}{15}\right) \times \left(\frac{1}{16}\right) \times \sum_{k=2}^{16}(-1)^{k}\binom{16}{k} e^{\left(20 \times \text{SINR} \times \left(\frac{1}{k}-1\right)\right)}$$

Where SINR = Signal-to-Interference-plus-Noise Ratio. BER should be between 0 and 1.

Propagation Loss:

Three different and mutually independent propagation phenomena influence the power of the received signal: ***path loss, shadowing and multipath fading***.

Shadowing:

Slow shadowing in wireless network is the attenuation caused by buildings or any obstacles between a transmitter and a receiver. In the model with shadowing, the shadowing value Xσ, typically defined in dB, is added to (or subtracted from) the average received power. *Xσ is a zero means Gaussian distributed random variable with standard deviation σ.*

The Probability Density Function (PDF) of the lognormal distribution is:

$$g(x) = \frac{1}{\sqrt{2\pi}\sigma x} \times e^{\left(\frac{(\ln x)^2}{2\sigma^2}\right)}$$

The default value for standard deviation is chosen as 5 dB.

Path Loss:

Pathloss is the reduction in power density of an electromagnetic wave as it propagates through space. Path loss may be due to many effects, such as free-space loss, refraction, diffraction, reflection, aperture-medium coupling loss, and absorption.

Path loss can be represented by the path loss exponent, whose value is normally in the range of 2 to 4, where 2 is for propagation in free space and 4 is for relatively loss environments. In NetSim, the default value for path loss exponent is taken as 2.

Path loss is usually expressed in dB. In its simplest form, the path loss can be calculated using the formula

$$L = 10\ n\ \log_{10}(d) + C$$

Where L is the path loss in decibels, n is the path loss exponent, d is the distance between the transmitter and the receiver, usually measured in meters, and C is a constant which accounts for system losses.

A simplified formula for the path loss between two isotropic antennas in free space:

$$L \text{ (in dBm)} = 20 \times \log_{10}\left(\frac{4\pi d}{\lambda}\right)$$

Where L is the path loss in decibels, λ is the wavelength and d is the transmitter-receiver distance in the same units as the wavelength.

Calculation of Received Power:

In general,

$$\binom{Received}{Power} = \binom{Transmitted}{Power} + \binom{Transmitter}{Gain} + \binom{Receiver}{Gain} - \binom{Path}{Loss}$$

The path loss model used is described in IEEE Standard 802.15.2-2003[B9], which stipulates a two-segment function with a path loss exponent of 2.0 for the first 8 m and then a path loss exponent of 3.3 thereafter. The formula given in IEEE Standard 802.15.2 is shown in Equation (E.1).

$$\text{pl(d)} = \begin{cases} pl(1) + 10\gamma_1 \log_{10} d \; d \geq 8m \\ pl(8) + 10\gamma_8 \log_{10}\left(\frac{d}{8}\right) d < 8m \end{cases}$$

Where, for 2.4 Ghz

pl(1) = 40.2 dBm

pl(8) = 58.5 dBm

γ_1 = 2, the path loss exponent for d ≤ 8m

γ_8 = 3.3, the path loss exponent for d > 8m

Reference: IEEE Standard 802.15.4 – 2006

Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs).

SINR, BER and Propagation models for 802.11a, b, g and n

Received Power Calculation

The received power can be expressed as

$[P_{rec}]_{dbm} = 10\log_{10}(P_t) + [G_T] + [G_R] + 20\log_{10}(\lambda/4\pi d_0) + 10\eta\log(d_0/d) + P_{shadow\ loss} + P_{fading}$

Where P_t = Power Transmitted (in mwatts)

G_T=Gain of the transmitting antenna (in dB)

G_R=Gain of the receiving antenna (in dB)

λ = Wavelength (in meters)

d_0= Reference distance (at which the path loss inherits free space path loss)

η = Path loss exponent (ranges between 2 to 5)

$P_{shadow\ loss}$ = Power due to Shadowing (in dB)

P_{fading} = Power due to Fading (in dB)

The code for calculating the received power is included in the file 802_11_PHY.c, path for the file is **NetSim Standard\src\Simulation\WLAN**.

Propagation Loss

Three different and mutually independent propagation phenomena influence the power of the received signal: ***path loss, shadowing and multipath fading.***

Path Loss

Path loss is the reduction in power density of an electromagnetic wave as it propagates through space. Path loss may be due to many effects, such as free reflection, aperture-medium coupling loss, and absorption.

Path loss can be represented by the path loss exponent, whose value is normally in the range of 2 to 4, where 2 is for propagation in free space.

In NetSim, the default value for path loss exponent is taken as 2.

Path loss is usually expressed in dB. In its simplest form, the path loss can be calculated using the formula

$$L=10\eta\log (d_0/d)$$

Where L is the path loss in decibels, η is the path loss exponent and d is the distance between transmitter and the receiver, usually measured in meters.

A simplified formula for the path loss between two isotropic antennas in free space:

$$L \text{ (in dBm)} = 20\log_{10} (4\pi d0/\lambda)$$

Where L is the path loss in decibels, _ is the wavelength and d is the transmitter-receiver distance in the same units as the wavelength.

The code for calculating the Path loss is included in the function fn_NetSim_WLAN_CalculateReceivedPower, which is present in the file 802_11_PHY.c, path for the file is **NetSim Standard\src\Simulation\WLAN**.

Shadowing

Slow shadowing in wireless network is the attenuation caused by buildings or any obstacles between a transmitter and a receiver. In the model with shadowing, the shadowing value X typically defined in dB, is added to (or subtracted from) the average received power *zero means Gaussian distributed random variable with standard deviation.* The default value for standard deviation is chosen as 5 dB.

The code for calculating the shadow loss is present in file 802_11_PHY.c, path for the file is **NetSim Standard\src\Simulation\WLAN**.

Fading

In wireless communications, fading is deviation of the attenuation affecting a signal over certain propagation media. The fading may vary with time, geographical position or radio frequency, and is often modelled as a random process.

In NetSim, the Rayleigh Fading, which follows Rayleigh Probability Distribution with mean of 1, is used. The code for calculating fading loss is present in the file 802_11_PHY.c, path for the file is **NetSim Standard\src\Simulation\WLAN**.

SINR Calculation

Analogous to the SNR used often in wired communications systems, the SINR is defined as the power of a certain signal of interest divided by the sum of the interference power (from all the other interfering signals) and the power of some background noise.

The interference power is the difference between the total power received by the receiver and the power received from one particular transmitter.

The background thermal noise in dBm at room temperature is given by:
P_{Noise} (in dBm) = $-174+10\log_{10}(\Delta f)$

Where Δf is the Bandwidth in Hertz. For 802.11a, b, g, Δf = 20 MHz, and for 802.11n, Δf = 20 MHz or 40 MHz.

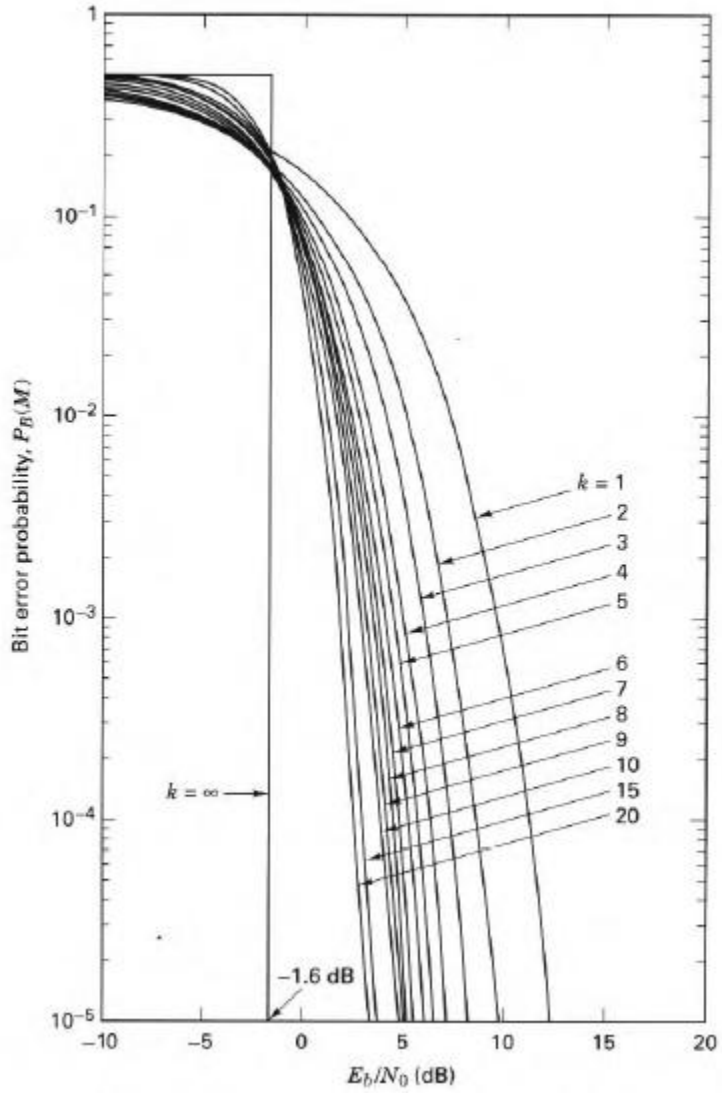
Therefore, SNR (in dBm) = Power Received (in dBm) - P_{Noise} (in dBm)

Bit Error Rate (BER) Calculation

The bit error rate (BER) is the number of bit errors divided by the total number of transferred bits during a studied time interval. The BER calculation has been done as a hash-table based on the BER curves for different modulation schemes.

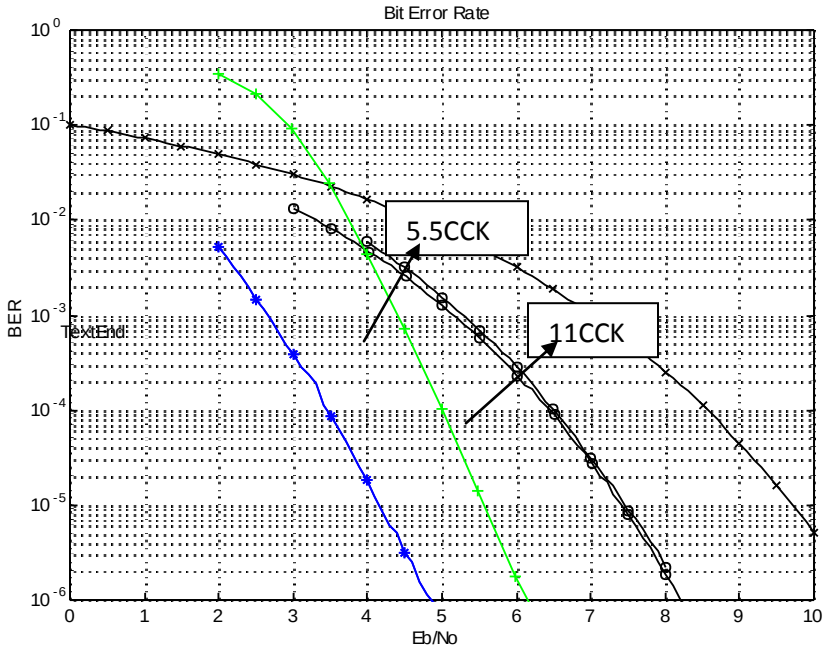
The code for calculating SNR and implementation of SNR vs. BER is included in the function fn_NetSim_WLAN_802_11x_BER of the file 802_11_x.c (where x=a, b, g and n), you can find it in the following path **NetSim Standard\src\Simulation\WLAN**.

SNR vs. BER curve for MPSK (where M=2k)



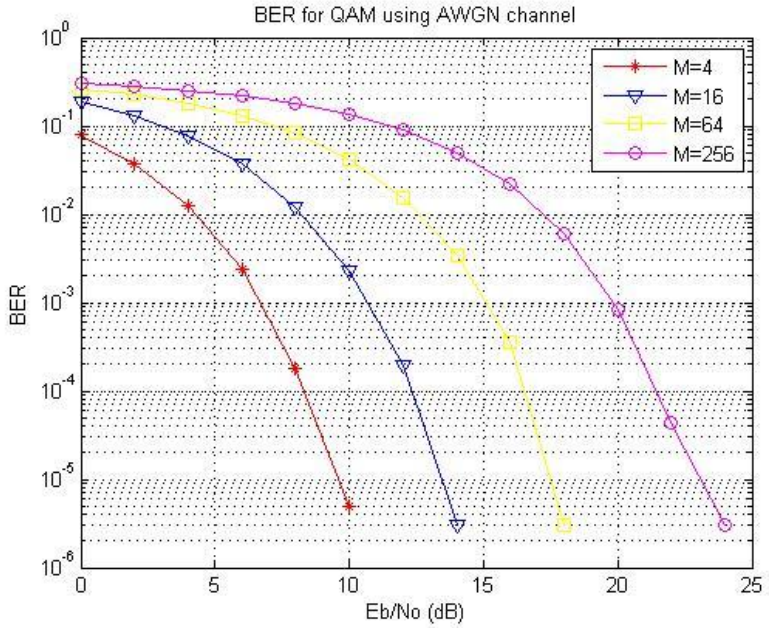
(Reference: From Page 221 Digital Communications by Bernard Sklar 2nd Edition)

SNR vs. BER curve for 5.5CCK and 11CCK



(Reference: Higher Rate 802.11b: Double the Data Rate Chris Heegard, Matthew Shoemake & Sid Schrum Doc: IEEE 802.11-00/091)

SNR vs. BER curve for MQAM



(Reference: BER Comparison of M-ary QAM by Mukthar Hussain MATLAB File Exchange)

SNR vs. BER Table followed in NetSim

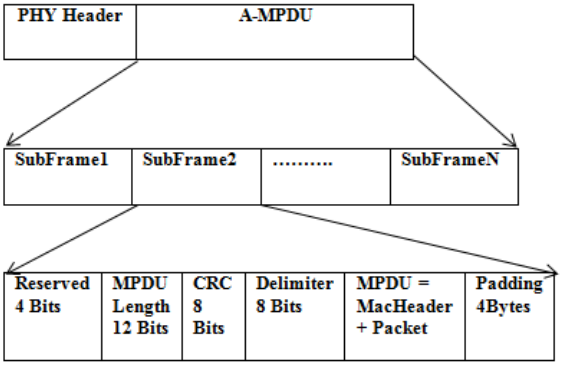
SNR(in dB)	BPSK	QPSK	16QAM	64QAM	5.5 CCK	11 CCK
0 to 0.5	7.00E-02	2.50E-01	1.50E-01	1.75E-01	4.50E-01	5.50E-01
0.5 to 1	6.00E-02	2.45E-01	1.35E-01	1.70E-01	4.50E-01	5.50E-01
1 to 1.5	5.00E-02	2.40E-01	1.15E-01	1.65E-01	1.75E-01	2.75E-01
1.5 to 2	4.00E-02	2.35E-01	1.00E-01	1.60E-01	1.75E-01	2.75E-01
2 to 2.5	2.50E-02	2.25E-01	9.50E-02	1.50E-01	9.50E-02	1.50E-01
2.5 to 3	1.50E-02	2.15E-01	9.00E-02	1.40E-01	9.50E-02	1.50E-01
3 to 3.5	1.00E-02	2.05E-01	8.00E-02	1.30E-01	4.50E-02	8.50E-02
3.5 to 4	9.50E-03	1.95E-01	7.00E-02	1.20E-01	4.50E-02	8.50E-02
4 to 4.5	9.00E-03	1.85E-01	6.00E-02	1.10E-01	9.50E-03	3.50E-02
4.5 to 5	7.00E-03	1.75E-01	5.00E-02	1.00E-01	9.50E-03	3.50E-02
5 to 5.5	5.00E-03	1.65E-01	4.00E-02	9.80E-02	6.00E-03	6.00E-03
5.5 to 6	2.50E-03	1.55E-01	3.00E-02	9.20E-02	6.00E-03	6.00E-03
6 to 6.5	1.50E-03	1.45E-01	2.00E-02	8.40E-02	1.50E-03	1.50E-03
6.5 to 7	8.50E-04	1.35E-01	1.00E-02	7.60E-02	1.50E-03	1.50E-03
7 to 7.5	7.00E-04	1.25E-01	9.50E-03	6.80E-02	6.00E-04	6.50E-04
7.5 to 8	2.00E-04	1.15E-01	8.00E-03	6.00E-02	6.00E-04	6.50E-04
8 to 8.5	9.00E-05	9.50E-01	6.50E-03	5.00E-02	7.00E-05	9.50E-05
8.5 to 9	6.00E-05	8.75E-02	5.00E-03	4.00E-02	7.00E-05	9.50E-05
9 to 9.5	3.00E-05	7.50E-02	3.50E-03	3.00E-02	7.00E-06	1.50E-06
9.5 to 10	1.50E-05	6.50E-02	2.00E-03	2.50E-02	7.00E-06	1.50E-06
10 to 11	1.00E-06	5.00E-02	1.00E-03	9.50E-03	8.00E-07	1.50E-07
11 to 12	1.00E-07	2.50E-02	7.00E-04	6.50E-03	9.50E-08	5.50E-08
12 to 13	1.00E-08	1.50E-02	3.00E-04	3.50E-03	2.50E-09	9.50E-09
13 to 14	1.00E-09	9.50E-03	1.00E-05	9.50E-04	1.00E-10	1.00E-10
14 to 15	1.00E-09	7.00E-03	1.00E-06	7.00E-04	1.00E-10	1.00E-10
15 to 16	1.00E-09	2.00E-03	1.00E-07	3.50E-04	1.00E-10	1.00E-10
16 to 17	1.00E-09	1.00E-06	1.00E-07	1.00E-05	1.00E-10	1.00E-10

WLAN 802.11n/802.11ac

IEEE 802.11n Standard

Mac aggregation and block acknowledgement are two important enhancements to 802.11n standard. In the aggregation scheme, several MPDU's (MAC Protocol Data Units) are aggregated in to a single A-MPDU (Aggregated MPDU).

The A-MPDU's are created before sending to PHY layer for transmission. The MAC does not wait for MPDU's before aggregation. It aggregates the already present packets in the queue to form an A-MPDU. The maximum size of A-MPDU is 65535 bytes. The maximum size of each MPDU is 4KB. In A-MPDU, each MPDU has a delimiter of 32bits at the beginning and padding ate the end. These padding bytes ensure that size of MPDU is a multiple of 4bytes.



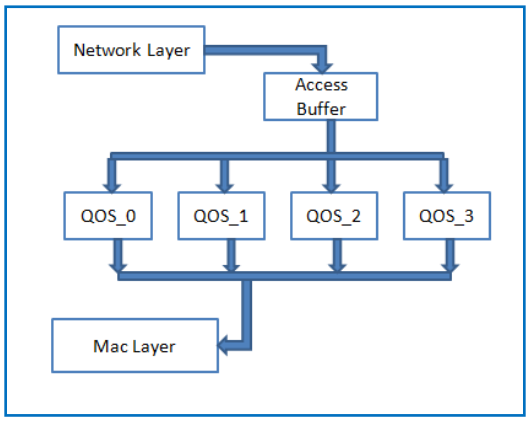
In 802.11n, a single block acknowledgement is sent for the entire A-MPDU. The block ack acknowledges each packet that is received. It consists of a bitmap(compressed bitmap) of 64bits or 8 bytes. This bitmap can acknowledge upto 64 packets, 1bit for each packet. The value of a bitmap field is 1 if respective packet is received without error else it is 0. Only the error packets are resent until a retry limit is reached. The number of packets in an A-MPDU is restricted to 64 since the size of block ack bitmap is 64bits.

Octets:2	2	6	6	2	2	8	4
Frame Control	Duration /ID	RA	TA	BARControl	BA Starting Sequence Control	BitMap	FCS

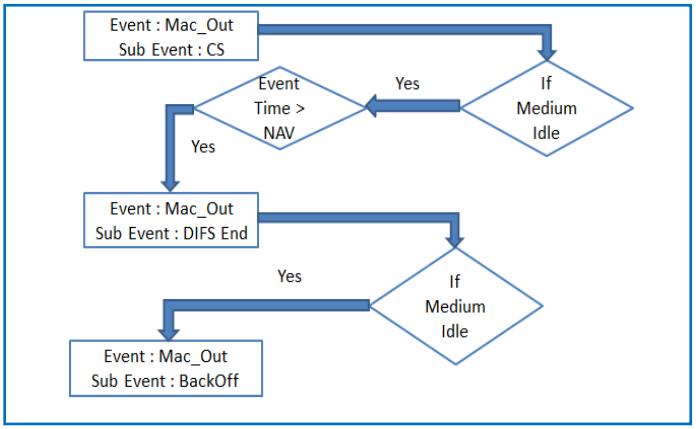
Block Ack Control Packet

Details of 802.11 n implementation in NetSim –

- NetSim aggregates packets in terms of numbers and not size.
- A user can vary the number of packets to aggregate by changing the appropriate parameters in the GUI.
- NetSim ignores the padding bytes added to the MPDU since its effect is negligible.
- NetSim aggregates packets to the same receiver id and not to the destination ID.



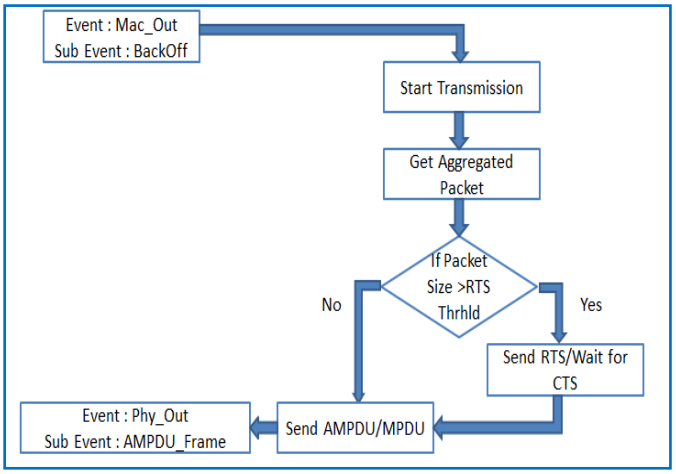
Packets arriving from the NETWORK Layer gets queued up in an access buffer from which they are sorted according to their priority in the respective QOS buffer according to the IEEE 802.11e standard. An event MAC_OUT with SubEvent CS (Carrier Sense – CSMA) is added to check if the medium is free



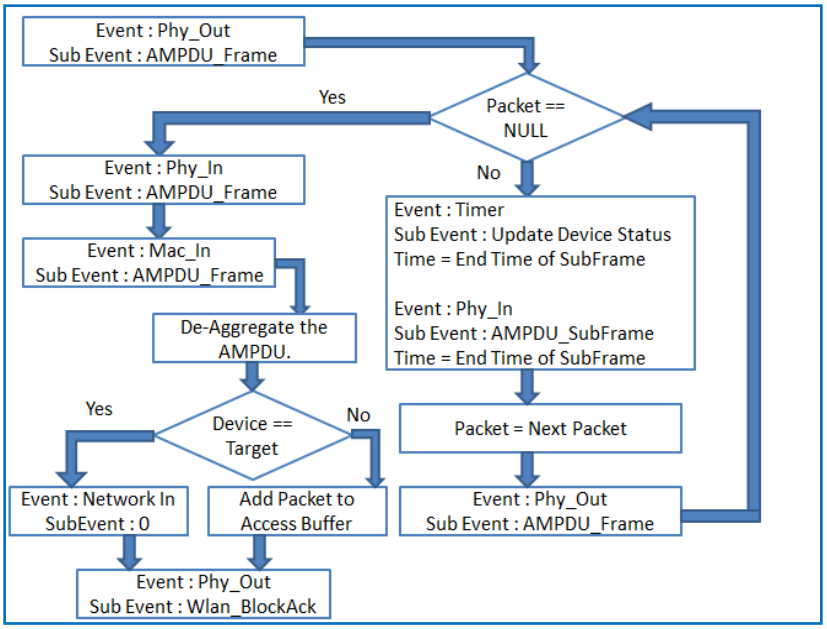
In CS, if the medium is free, then the NAV is checked. This is enabled if RTS/CTS mechanism is enabled which can be done so by adjusting the RTS Threshold. If the Present_Time>NAV, then an Event MAC_OUT with SubEvent DIFS End is added at the time Present_Time + DIFS time.

The medium is checked at the end of DIFS time period and a random time BackOff is calculated based on the Contention Window (CW). An Event MAC_OUT with SubEvent Backoff is added at time Present_Time + BackOff Time.

Once Backoff is successful, NetSimstarts the transmission process wherein it gets the aggregated packet from the QOS buffer and stores it in the Retransmit buffer. If the A-MPDU size is > RTS Threshold, then it enables RTS/CTS mechanism which is an optional feature.



NetSim sends the packet by calling the PHY_OUT Event with SubEventAMPDU_Frame. Note that the implementation of A-MPDU is in the form of a linked list.



Whenever a packet is transmitted, the medium is made busy and a Timer Event with SubEvent Update Device Status is added at the transmission end time to set the medium again as idle.

Events PHY_OUT SubEvent AMPDU_SubFrame, Timer EventSubEvent Update Device Status and Event PHY_IN SubEvent AMPDU_SubFrame are added in succession for each MPDU(Subframe of the aggregated packet). This is done for collision calculations. If two stations start transmission simultaneously, then some of the SubFrames may collide. Only those collided SubFrames will be retransmitted again. The same logic is followed for an Errored packet. However, if the PHY header (the first packet) is errored or collided, the entire A-MPDU is resent.

At the receiver, the device de-aggregates the packet in the MAC Layer and generates a block ACK which is sent to the transmitter. If the receiver is an intermediate node, the de-aggregated packets are added to the access buffer of the receiver in addition to the packets which arrive from Network layer. If the receiver is the destination, then the received packets are sent to the Network layer. At the transmitter side, when the device receives the block acknowledgement, it retransmits only those packets which are errored. The rest of the packets are deleted from the retransmit

buffer. This is done till all packets are transmitted successfully or a retransmit limit is reached after which next set of packets are aggregated to be sent.

802.11ac MAC and PHY Layer Implementation

Improvements in 802.11ac compared to 802.11n

Feature	802.11n	802.11ac
Spatial Streams	Up to 4 streams	Up to 8 streams
MIMO	Single User MIMO	Multi-User MIMO
Channel Bandwidth	20 and 40 MHz	20, 40, 80 and 160 MHz (optional)
Modulation	BPSK, QPSK, 16QAM and 64QAM	BPSK, QPSK, 16QAM, 64QAM and 256QAM (optional)
Max Aggregated Packet Size	65536 octets	1048576 octets

MAC layer improvements include only the increment of number of aggregated packets from 64 to 1024. The MCS index for different modulation and coding rates

MCS Index	Modulation	Code Rate
0	BPSK	1/2
1	QPSK	1/2
2	QPSK	3/4
3	16QAM	1/2

4	16QAM	3/4
5	64QAM	2/3
6	64QAM	3/4
7	64QAM	5/6
8	256QAM	3/4
9	256QAM	5/6

Receiver sensitivity for different modulation schemes in 802.11ac (for a 20MHz Channel bandwidth)

MCS Index	**Receiver Sensitivity (in dBm)**
0	-82
1	-79
2	-77
3	-74
4	-70
5	-66
6	-65
7	-64
8	-59
9	-57

Number of subcarriers for different channel bandwidths

PHY Standard	**Subcarriers**	**Capacity relative to 20MHz in 802.11ac**
802.11n/802.11ac 20MHz	Total 56, 52 Usable (4 pilot)	x1.0
802.11n/802.11ac 40MHz	Total 114, 108 Usable (6 pilot)	x2.1
802.11n/802.11ac 80MHz	Total 242, 234 Usable (8 pilot)	x4.5
802.11n/802.11ac 160MHz	Total 484, 468 Usable (16 pilot)	x9.0

Now with the knowledge of MCS index and bandwidth of the channel data rate is set in the following manner

Step1: Get the number subcarriers that are usable for the given bandwidth of the medium.

Step2: Get the Number of Bits Per Sub Carrier (NBPSC) from selected MCS

Step3: Number of Coded Bits Per Symbol (NCBPS) = NBPSC*Number of Subcarriers

Step4: Number of Data Bits Per Symbol (NDBPS) = NCBPS*Coding Rate

Step5: Physical level Data Rate = NDBPS/Symbol Time (4micro sec for long GI and 3.6 micro sec for short GI)

Real Time - Frames Capture (Available in Academic Version only)

Description: Monitor the traffic using Frames capture.

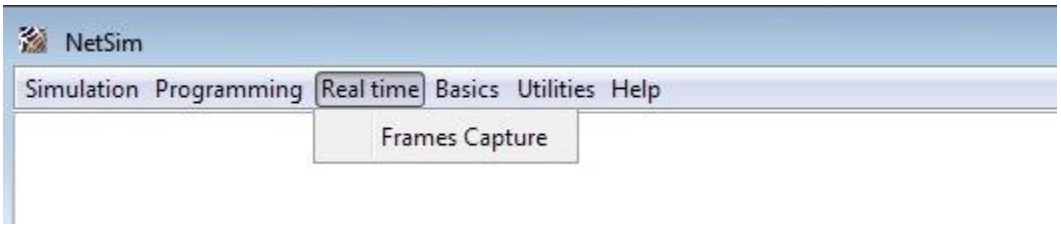
To capture & analyze packets in a network we require active systems connected in the network and a network monitoring tool. The network monitoring tool observes the network for data movement, and as the tool recognizes data movement, it will copy the data (even if the data is not destined for the machine running the monitoring software) and analyze it. The analyses of data include the correctness of the data, the data payload, overheads, the time of capture and also the protocols involved. Based on the data analyses the tool calculates the utilization and effective utilization of the network observed.

Procedure:

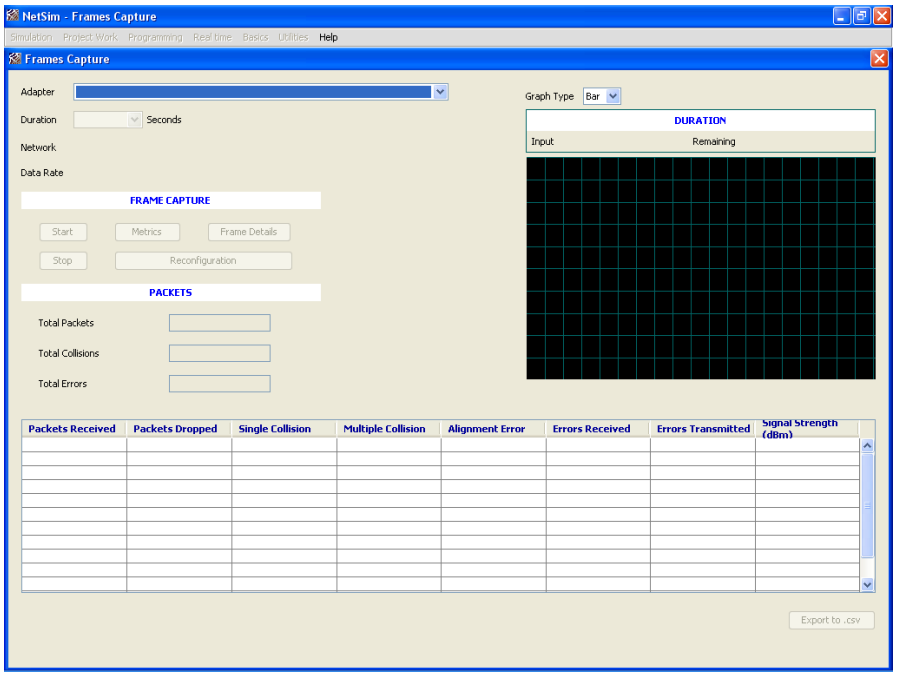
(1) Connect two systems by using a hub (for testing a single broadcast domain) or via a Switch

(2) To begin with the experiment, open NetSim

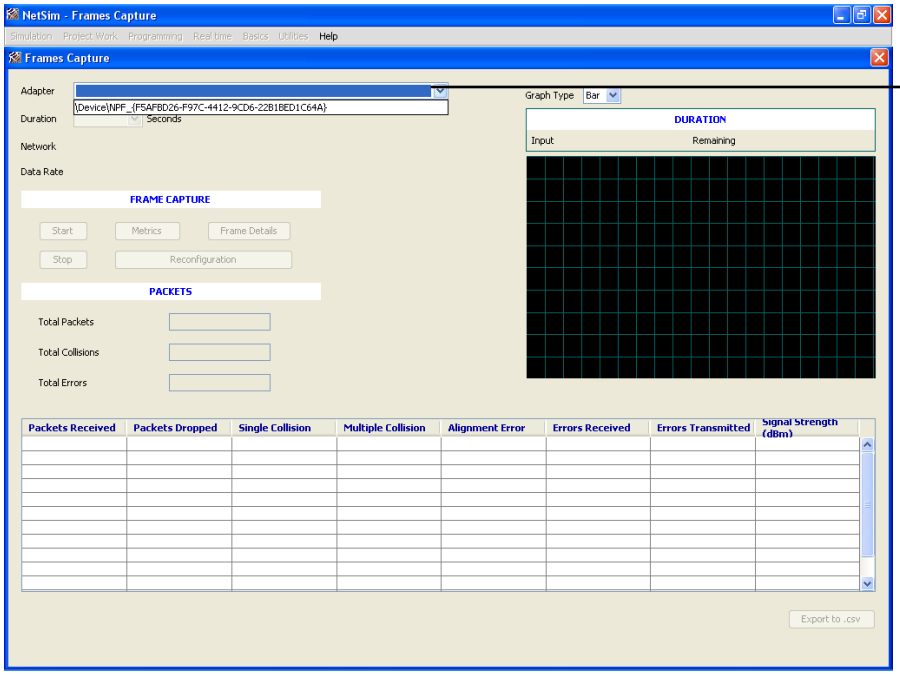
(3) Click on Real Time, select Frame Capture. The frame capture environment is now open.



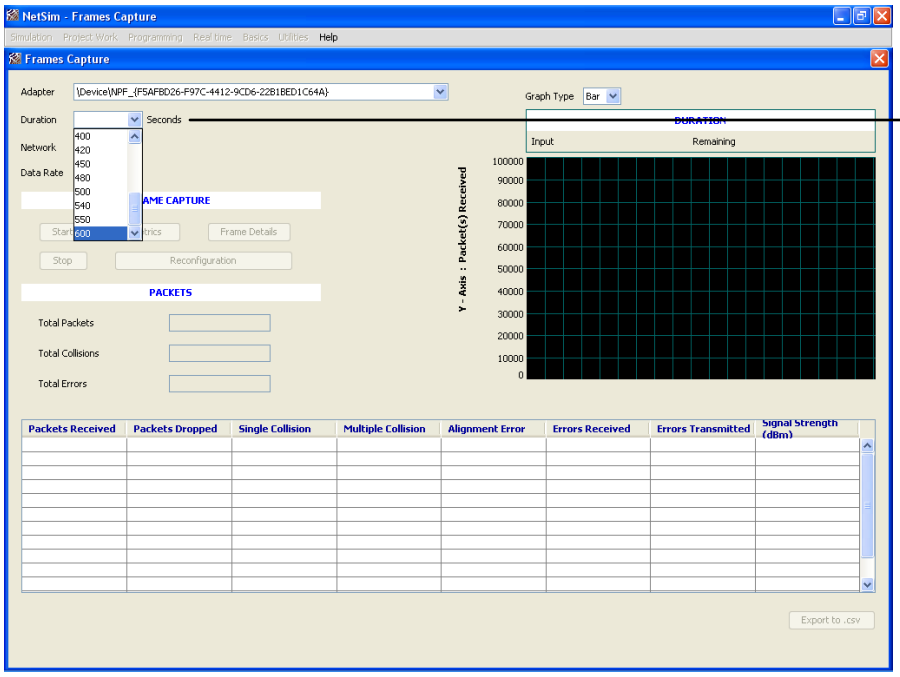
The below figure shows the NetSim frame capture environment



(4) Select the adapter



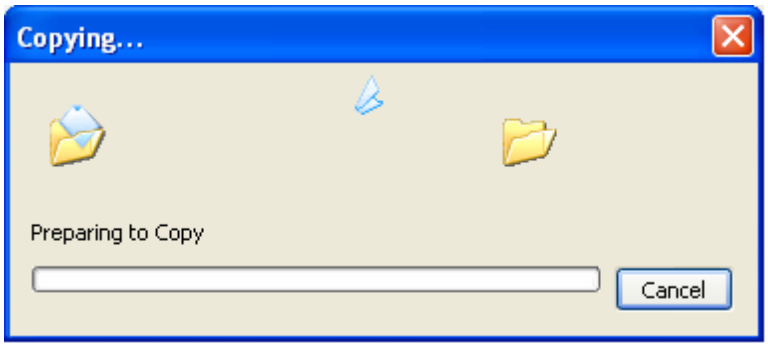
(5) Select how long the traffic is to be monitored



Select the frame capture duration

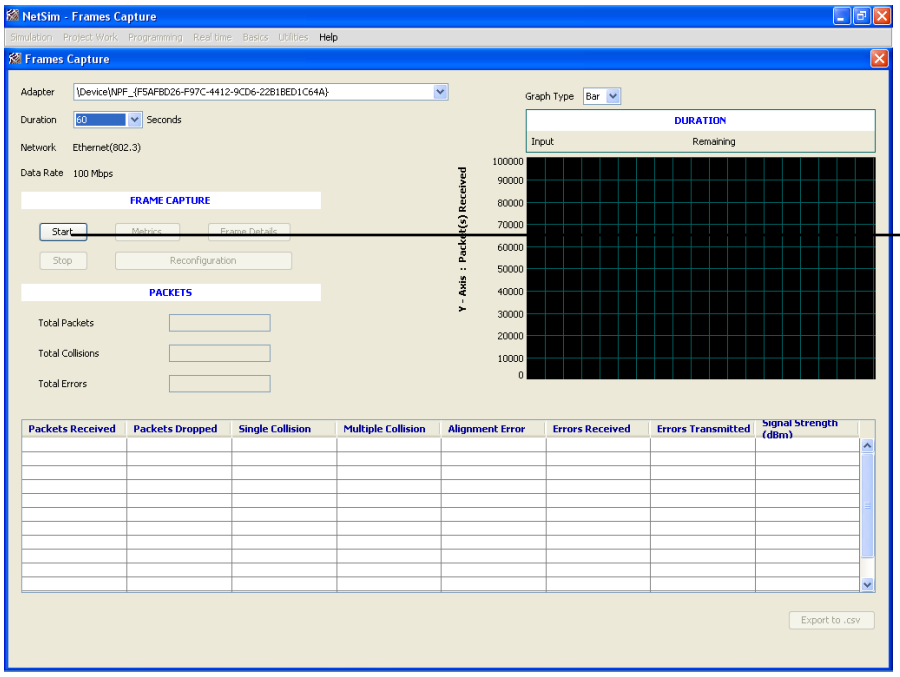
(6) Copy a reasonably large file from one system to the other system.

Note: The file should be large enough for the transfer to take sufficient time (>20 seconds) for the tool to analyze the network accurately.

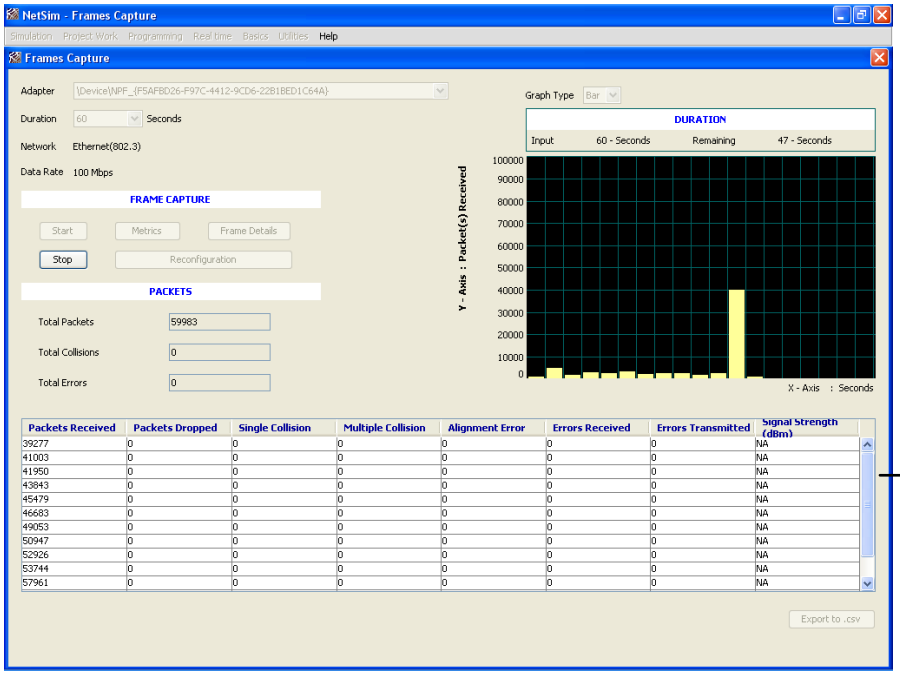


Transfer data from one machine to another

(7) As the file is getting transferred between systems start the traffic monitor by clicking the 'Start' button.

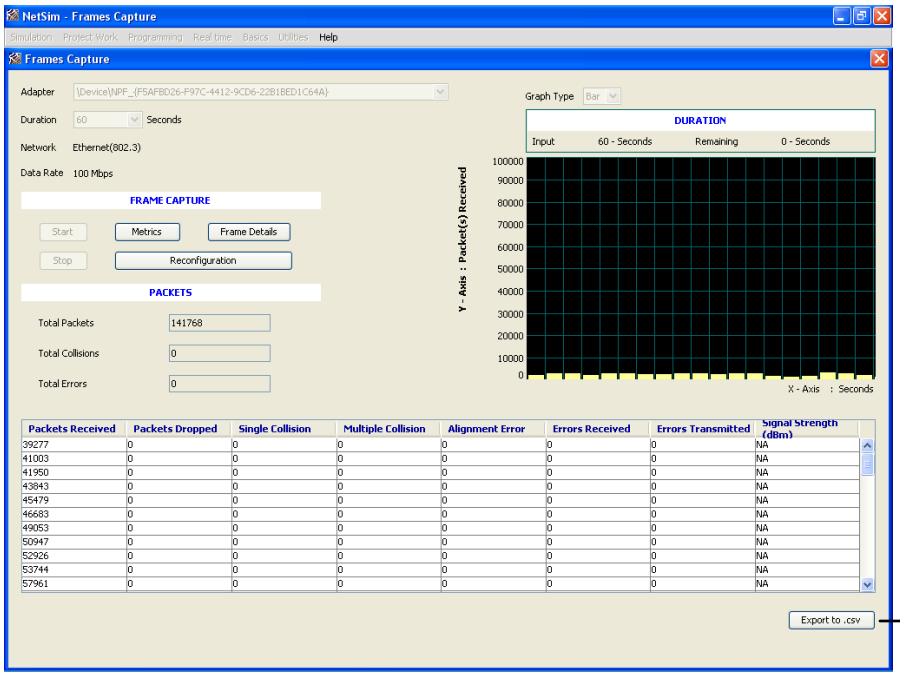


Click the Start button

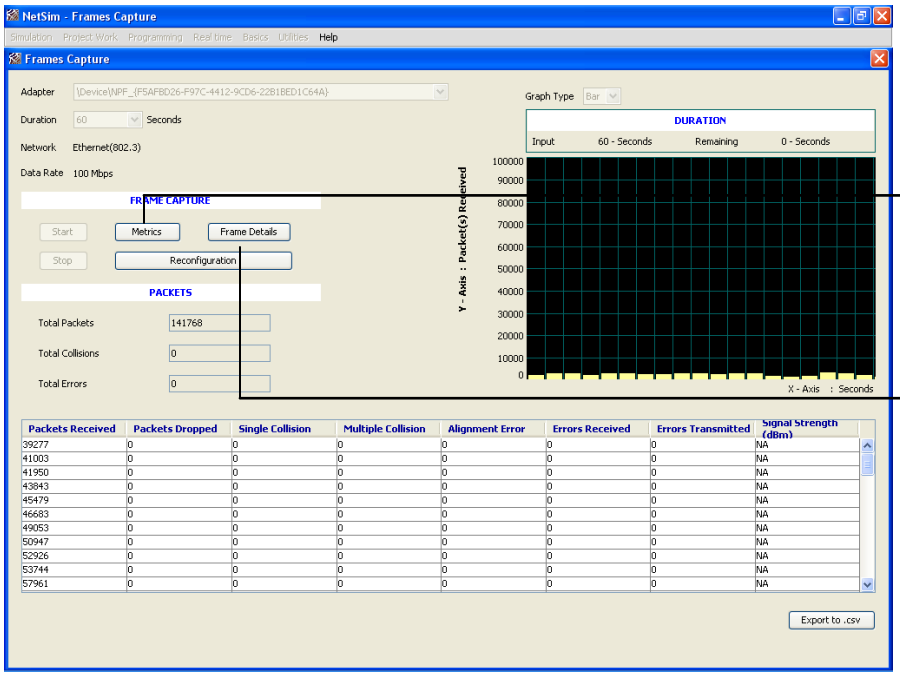

NetSim - Frames Capture
Simulation Project Work Programming Real time Basics Utilities Help
Frames Capture
Adapter \Device\NPF_{F5AFBD26-F97C-4412-9CD6-22B1BED1C64A}
Duration 60 Seconds
Network Ethernet(802.3)
Data Rate 100 Mbps
FRAME CAPTURE
Start
Metrics
Frame Details
Stop
Reconfiguration
PACKETS
Total Packets 59983
Total Collisions 0
Total Errors 0
Graph Type Bar
DURATION
Input 60 - Seconds
Remaining 47 - Seconds
Y - Axis : Packet(s) Received
X - Axis : Seconds
Packets Received
Packets Dropped
Single Collision
Multiple Collision
Alignment Error
Errors Received
Errors Transmitted
Signal Strength (dBm)
39277 0 0 0 0 0 0 NA
41003 0 0 0 0 0 0 NA
41950 0 0 0 0 0 0 NA
43843 0 0 0 0 0 0 NA
45479 0 0 0 0 0 0 NA
46683 0 0 0 0 0 0 NA
49053 0 0 0 0 0 0 NA
50947 0 0 0 0 0 0 NA
52926 0 0 0 0 0 0 NA
53744 0 0 0 0 0 0 NA
57961 0 0 0 0 0 0 NA
Export to .csv

Data from the network getting captured

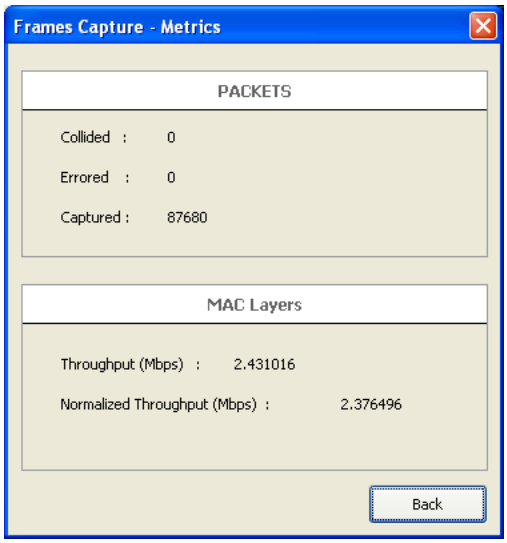
(8) After the duration time end click ‘Export to .csv’ button and note the values.



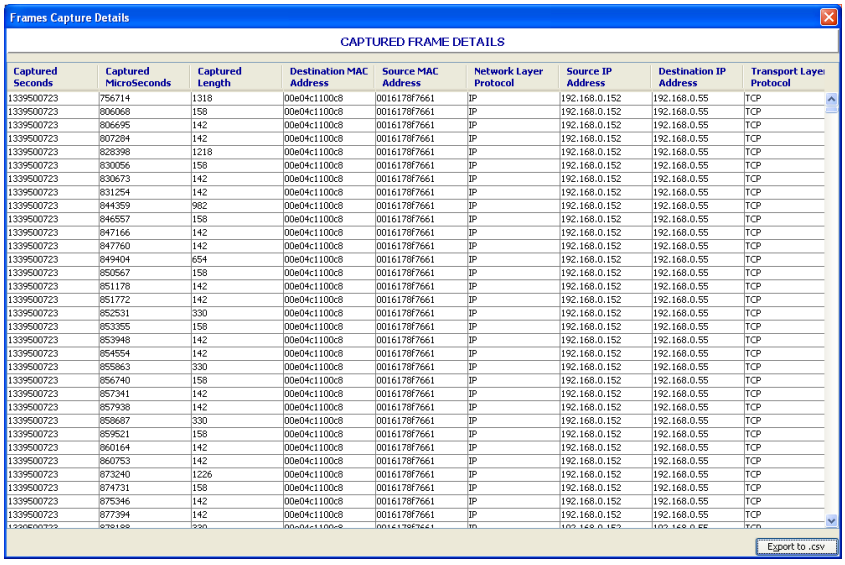
The details of the frame and the performance of the network can be viewed by clicking ‘Metrics’ and ‘Frame Details’ button.



Metrics Screen:



Frame Details Screen:



(9) Repeat the same procedure by increasing the number of systems in the network, and also increase the number of nodes transferring data in the network.

Inference:

Using Netpatrol one can analyze the throughput of the file transfer in the network. Furthermore you can drill down to each packet and get information relating to packet length, source IP address, Destination IP address, Source MAC Address, Destination MAC Address etc.

Programming Menu (Available only in Academic version)

This menu contains network programming exercises. Run down the menu and select the desired programming exercise. The programs available are as follows,

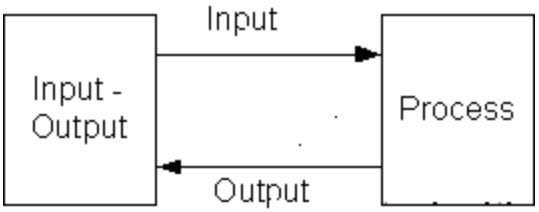
- Assignments of Sites to Concentrator
- Address Resolution Protocol
- Cryptography
    - Substitution
    - Transposition
    - XOR
    - Advanced
        - Data Encryption Standard
        - RSA
        - Wired Equivalent Privacy
- Distance Vector Routing
- Dynamic Host Configuration Protocol
- Error Correcting Code
    - Hamming Code
- Error Detecting Codes
    - Cyclic Redundancy Check
    - Longitudinal Redundancy Check
- Framing Sequence
    - Bit Stuffing
    - Character Stuffing
- Generic Cell Rate Algorithm
    - Virtual Scheduling Algorithm
- IPV4 Addressing
    - Address Mask
    - Binary Conversion
    - Classless InterDomain Routing
    - Network Address

    - Special Addresses
    - Subnetting
- IPV6 Addressing
    - EUI - 64 Interface Identifier
    - Host Addresses
    - Subnetting
- Leaky Bucket Algorithm
- Multi Level Multi Access
- Multiple Access Technology
    - CDMA
    - TDMA
    - OFDMA
- PC to PC Communication
    - Socket Programming
    - Chat Application
- Scheduling
- Shortest Path
- Sliding Window Protocol
- Sorting Techniques
- Spanning Tree
- Transmission Flow Control

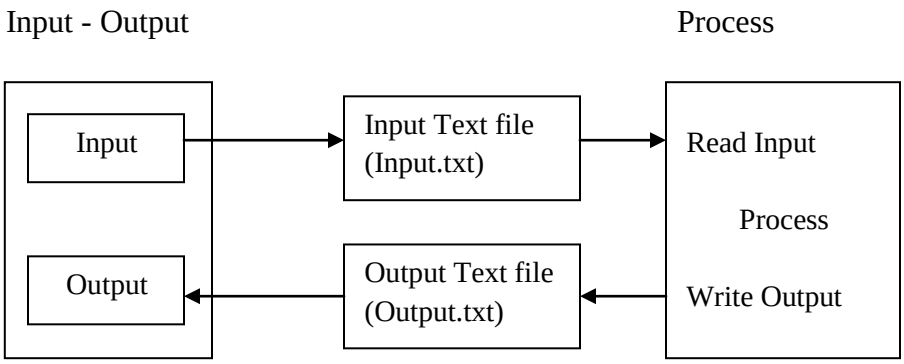
NetSim's Programming Lab has been designed to provide hands - on network programming skills to students. The labs come with a GUI where students can first run the experiment in "**Sample mode**" and subsequently link their own code and visualize it's working. Programs can be written in C and the executable can be linked to NetSim.

Architecture

The following **Architecture** is applicable for all the exercises under the **Programming** menu. Each exercise has two modules



Using the **Input - Output** module inputs are given and output is viewed. The working of the concept/algorithm is done in the process module. The link between the **Input - Output** module and process module is as follows



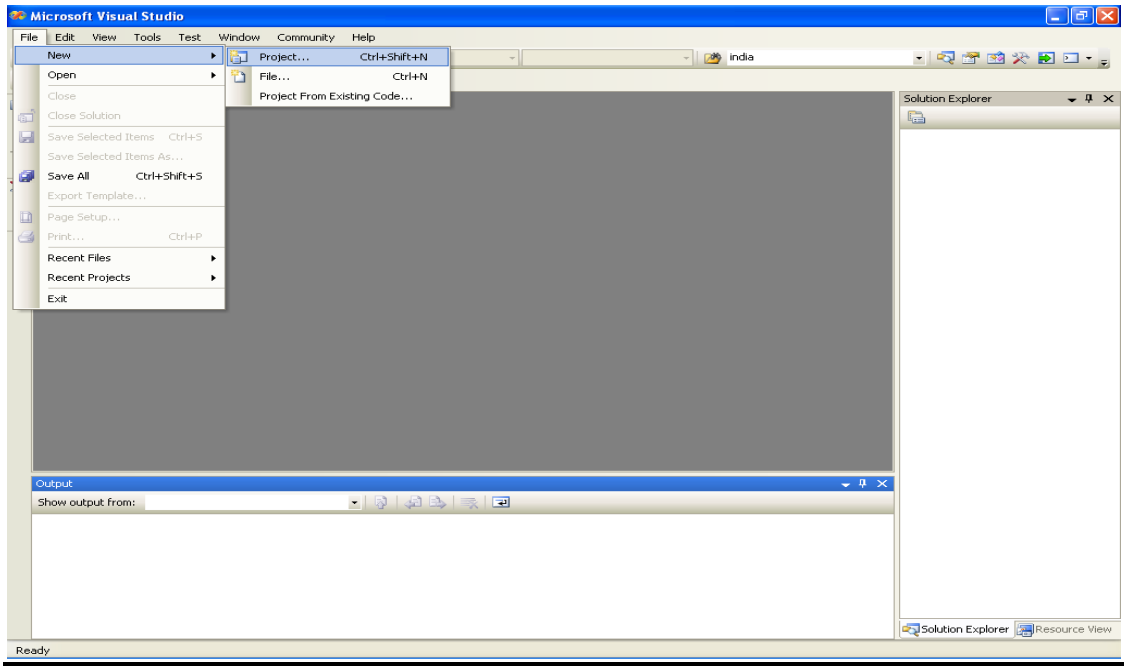
Process is the module for which the user has to write and link the written code when using the user mode.

The code can be written either by C or C++, the executable file created should be linked to the software.

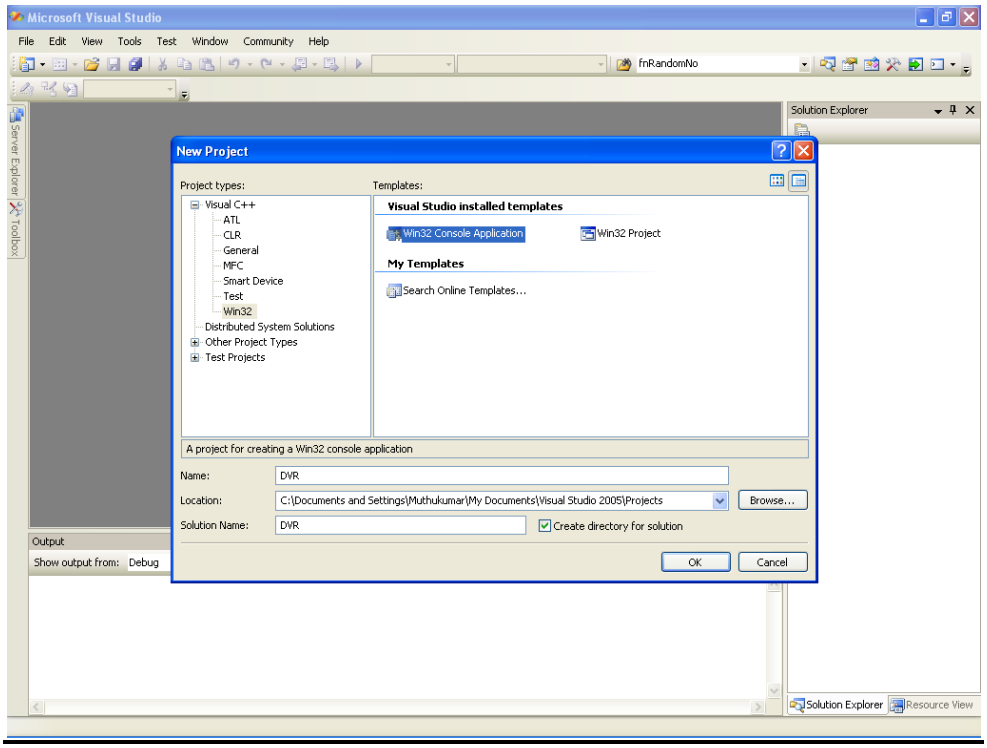
The input and output file format should be as required, as they form the link between the software and the user executable file.

Creating .exe using Visual Studio

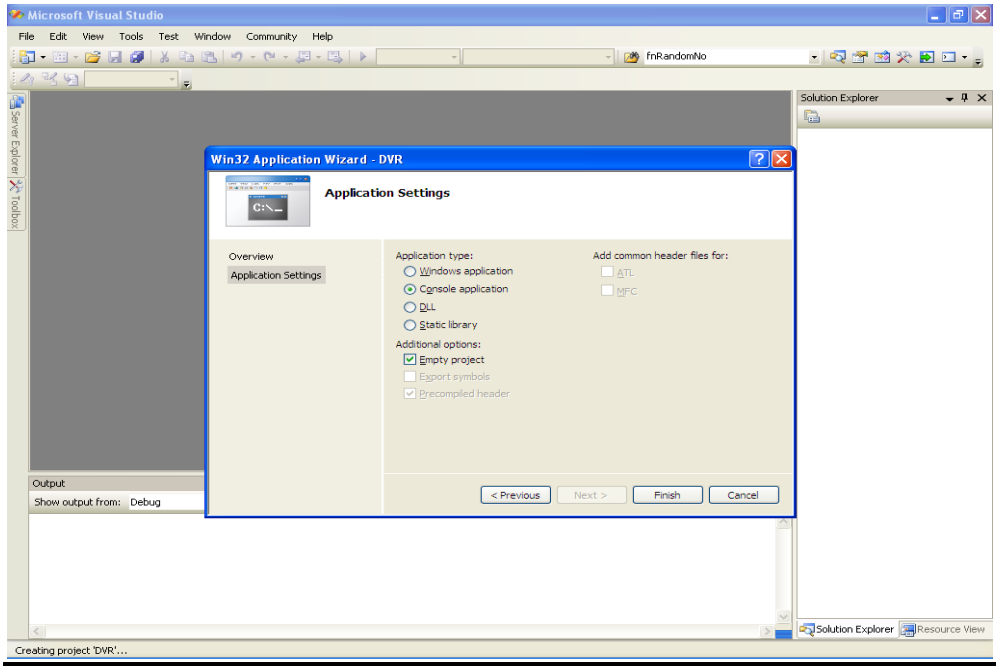
1. Select File→New→Project



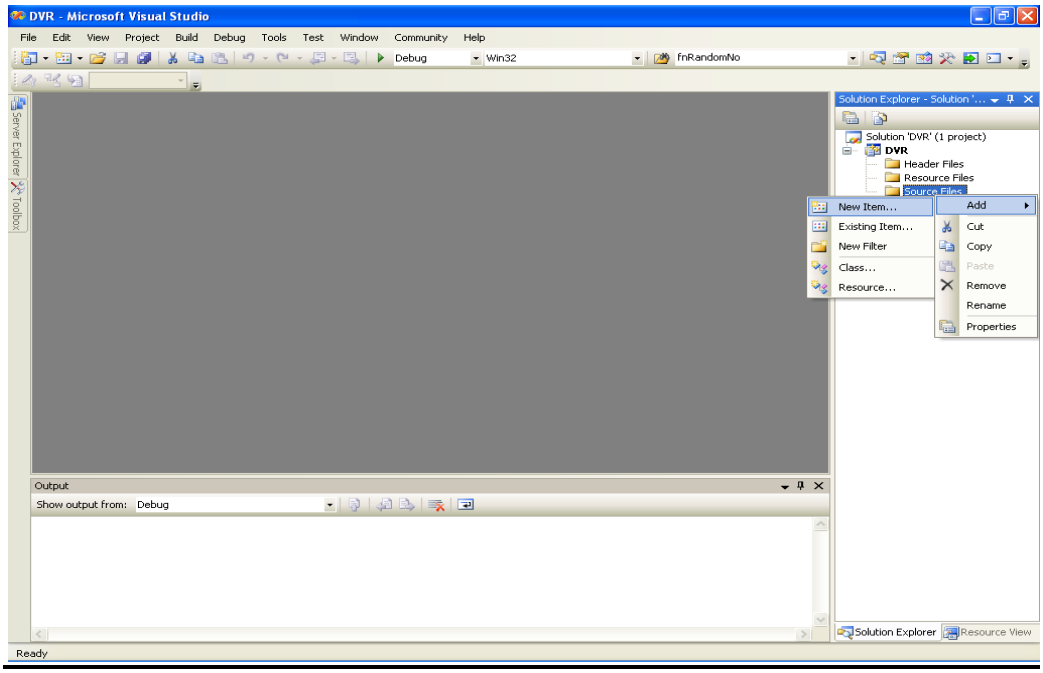
2. Select **Win32Console Application.** Name the project and select location to save the project and then click OK button.



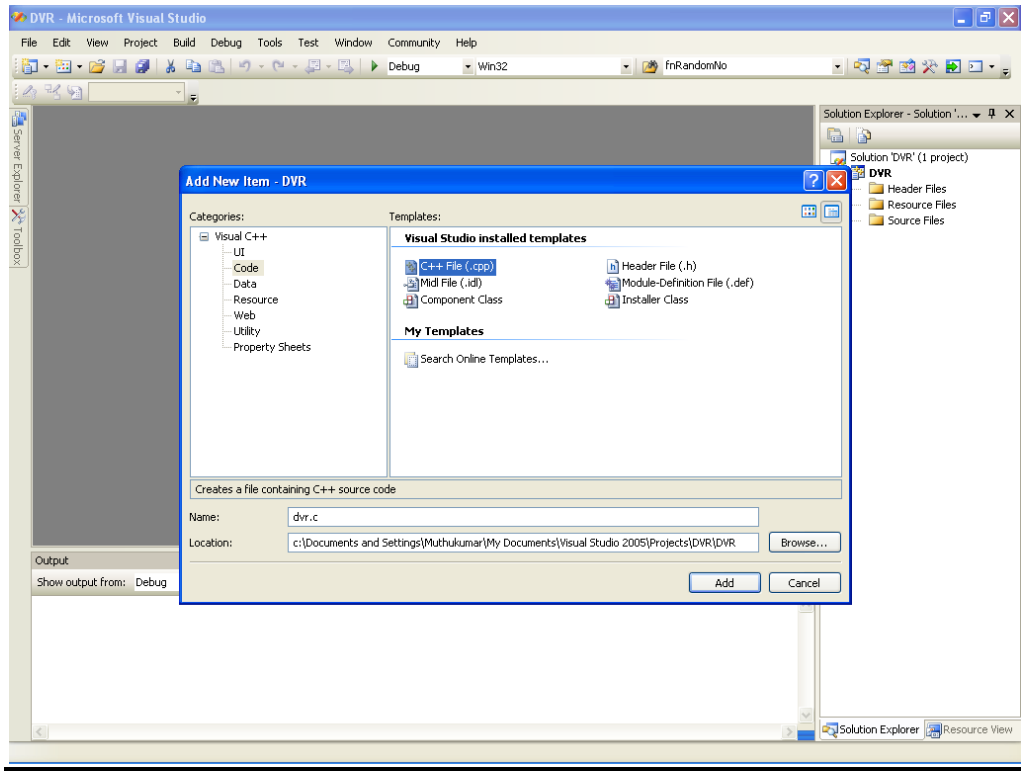
3. Click Next→ Check Whether Console application is selected or not. If selected, then select Empty Project, otherwise select Console application and Empty Project and finally click Finish button.



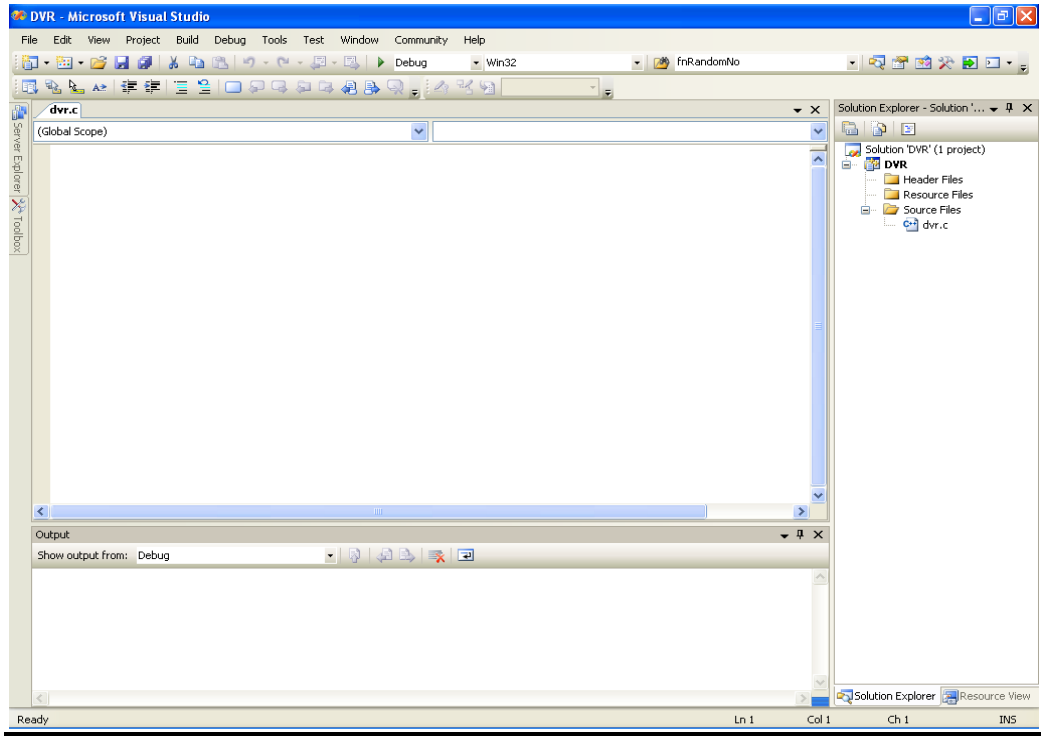
4. Add source codes to the project. Right click on the Source Files→Add→New Item



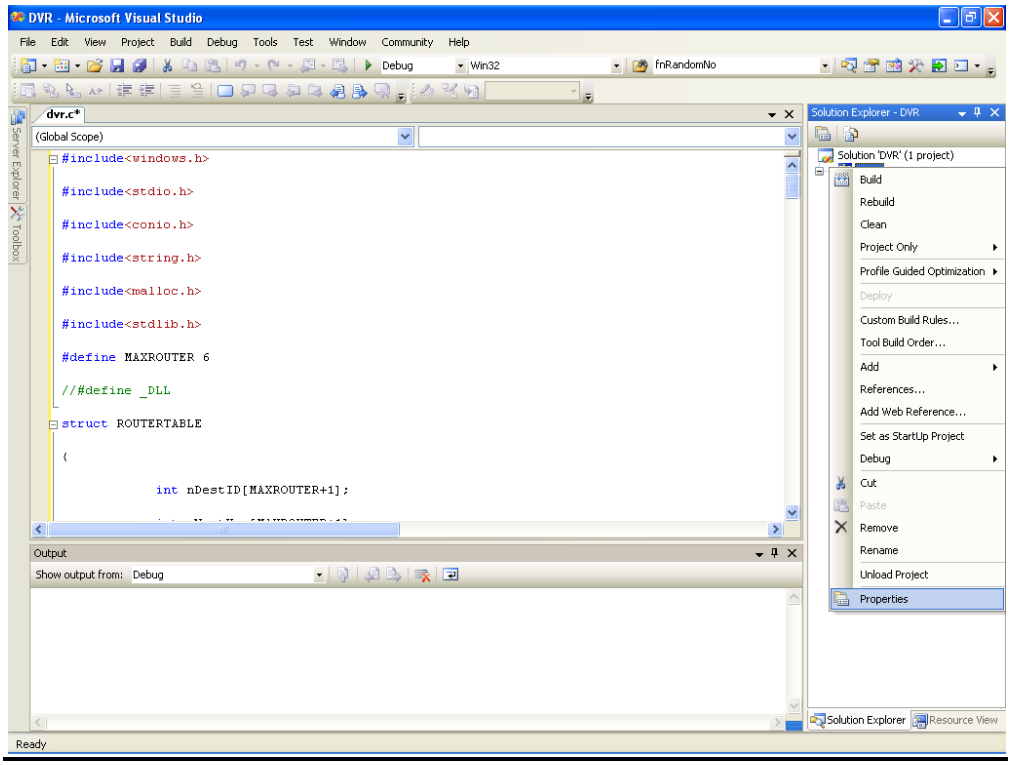
5. Select c++ File (.cpp) and name the file with extension of .c



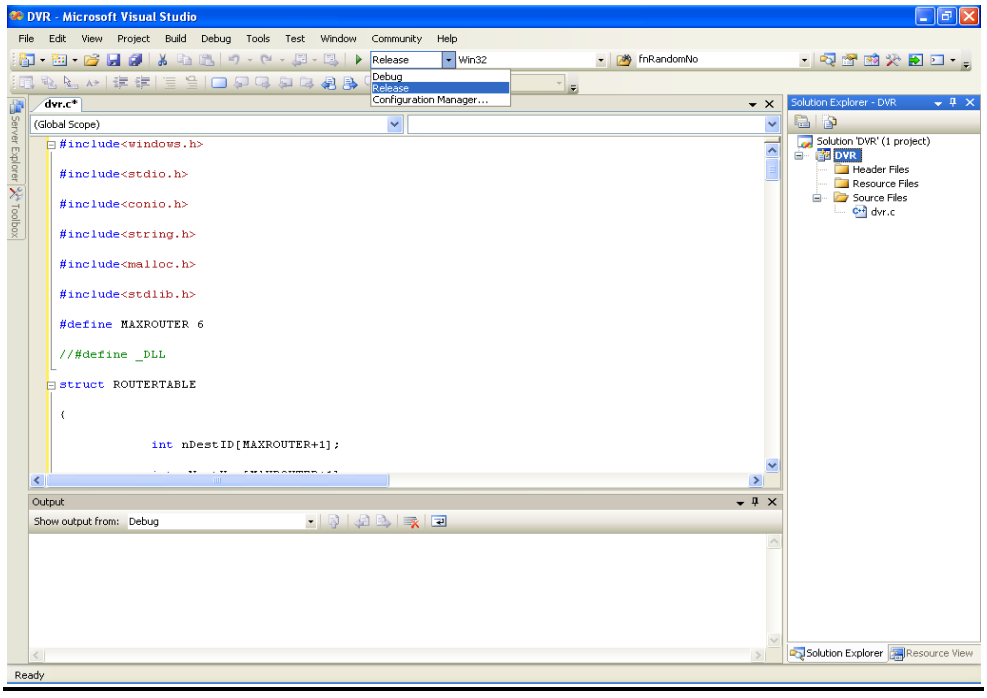
6. Now Source file is created



7. Copy and paste the source code.

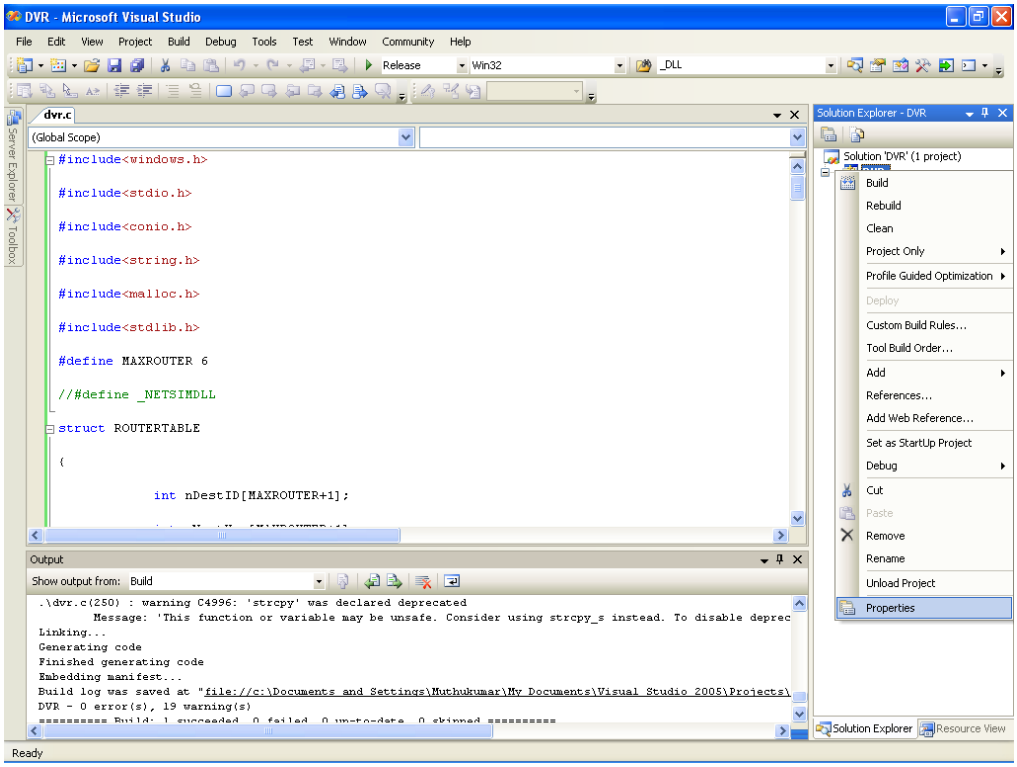


8. Select the mode for creating the exe i.e., debug or release mode by choosing the required option as shown in the figure. The preferred setting is debug mode.

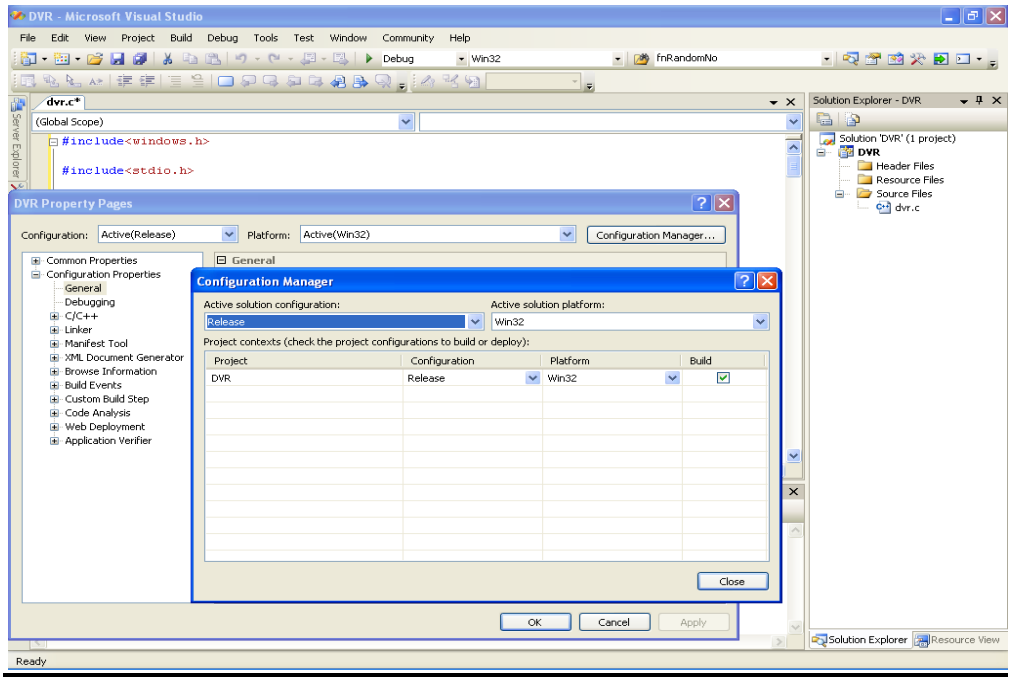


Alternate way to create an exe in Debug/Release Mode

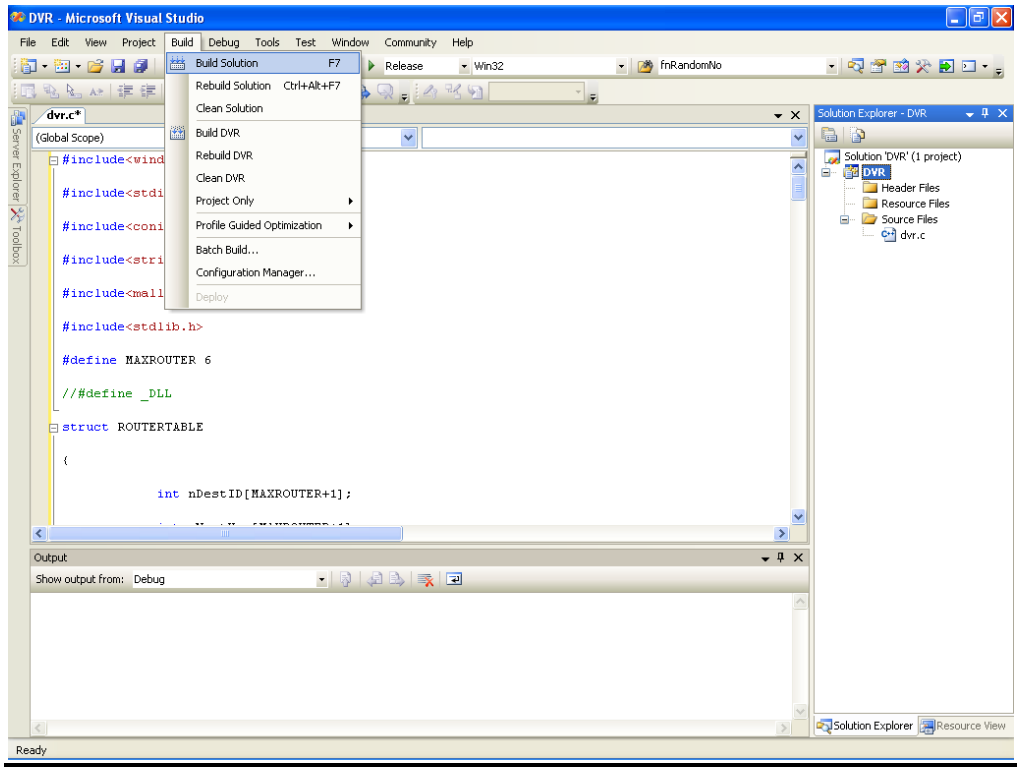
a. Right click on the Project File→Properties



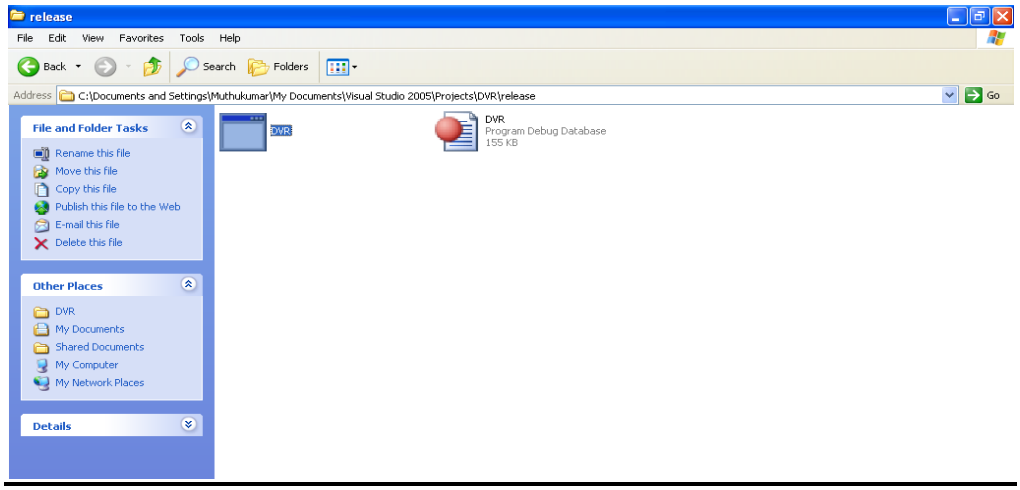
b. Click Configuration Manager ,a new dialog box will appear. Select Active Solution Configuration as Debug/Release →close and Finally OK.



11. To build the Solution,Select Build Menu →Build Solution



12. Now, Exe is created in the Project Folder as shown below.

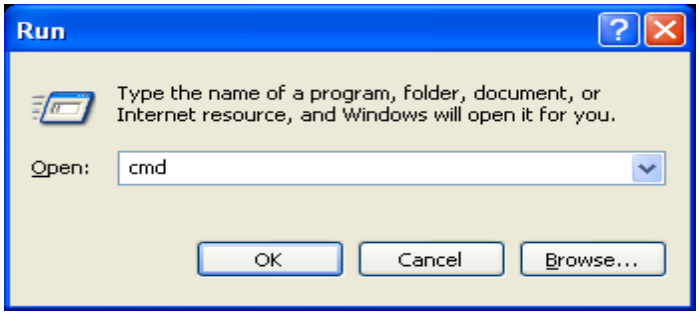


Creating .exe using GCC

- **By using Command Prompt:**

  C / C++ files can be created using any editor. Ex: Notepad

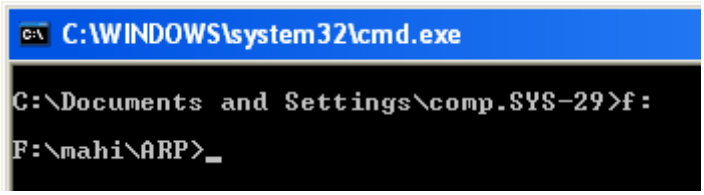
  Once C / C++ file is ready go to command prompt using Start → Run.



There are three cases for creating Exe (.exe) file using GCC.

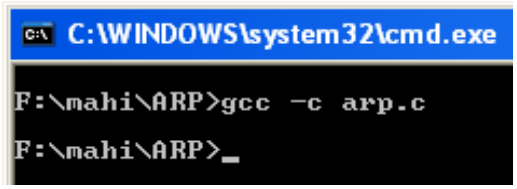
Case 1: C programs

1. Set the path as C program path



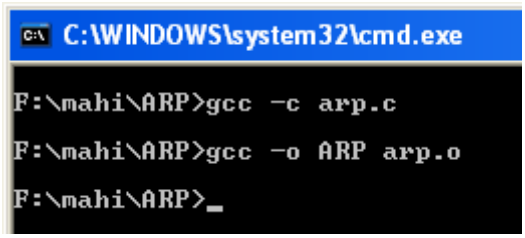
2. Create Output file (.o file) using the command

   gcc - C Filename.c



3. Create exe file (.exe file) using the command

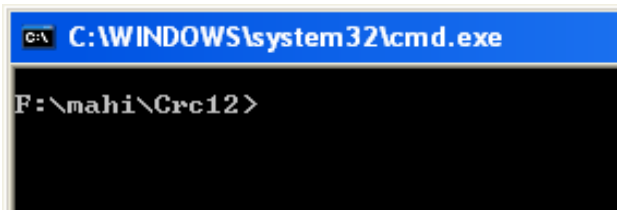
   gcc -o filename filename.o



4. Once exe file is created link that exe file with NetSim → Programming → User mode

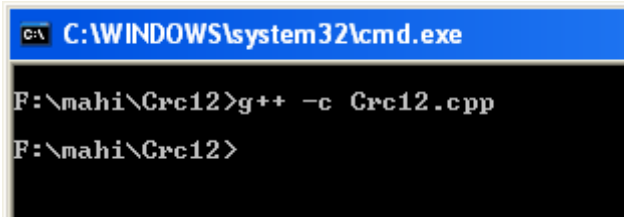
Case 2: C++ programs

1. Set the path as C++ program path.



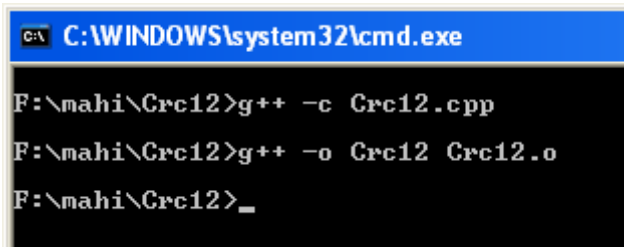
2. Create output file (.o file) using the command.

g++ -C Filename.c



3. Create exe file (.exe file) using the command

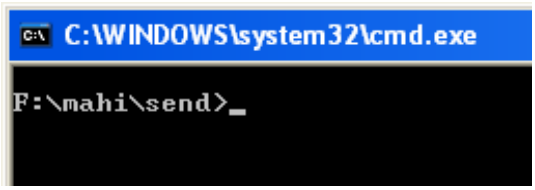
g++ -o filename filename.o



4. Once exe file is created link that exe file with NetSim → Programming → User mode

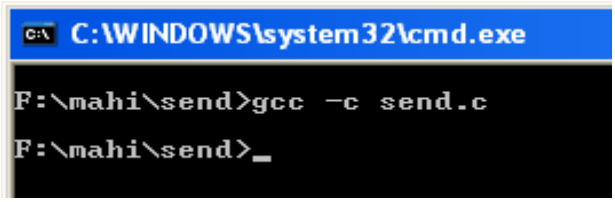
Case 3: Socket programs

1. Set the path to Socket program location.



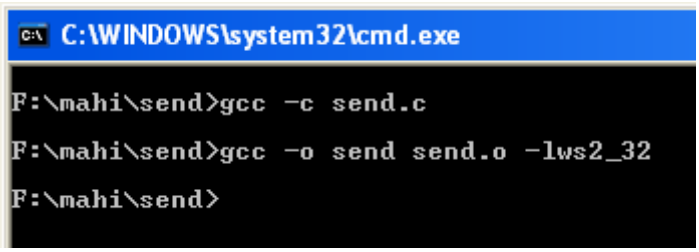
2. Create output file (.o file) using below command.

gcc -C Filename.c



3. Create exe file (.exe file) using below command

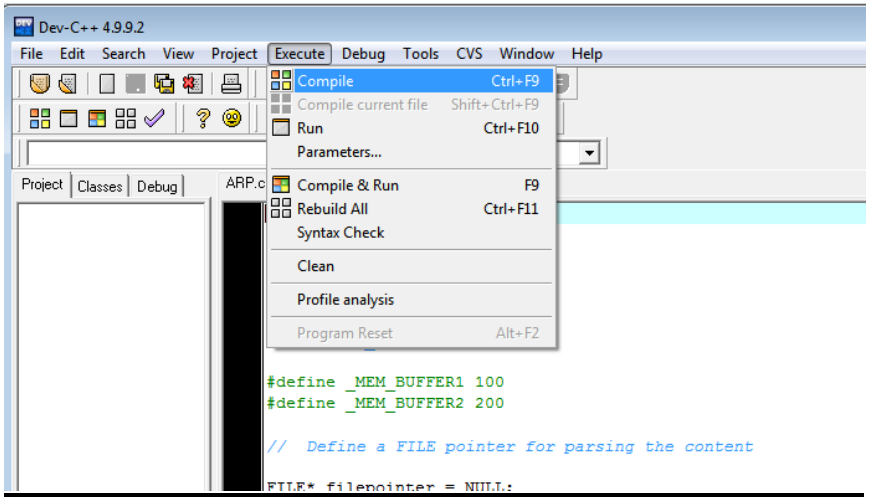
gcc -o send send.o -lws2_32



4. Once exe file is created link that exe file with NetSim → Programming → User mode

Creating .exe using Dev C++

1. Open the C source code file in Dev C++. Then go to Execute → Compile.



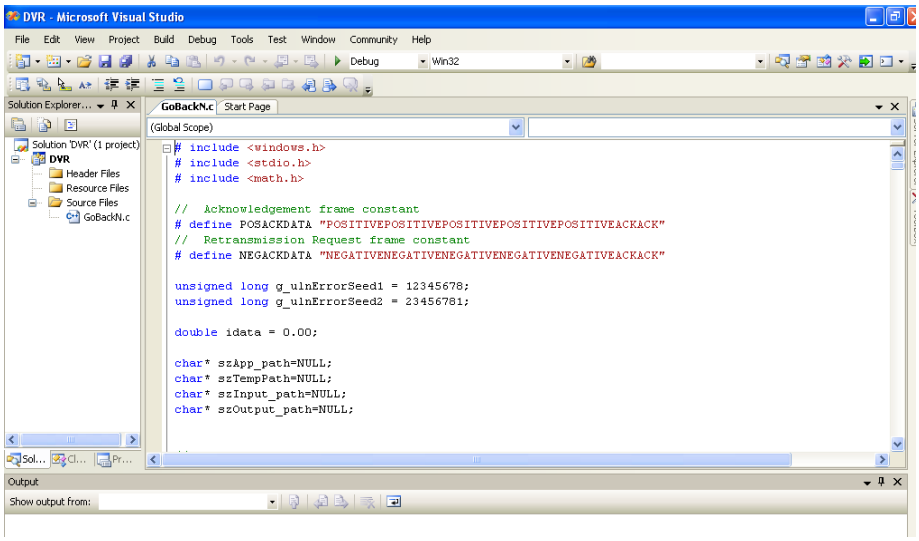
Dev C++ will create the exe file in the same location where the C source code file is located

2. Once exe file is created , link that exe file with NetSim → Programming → User mode

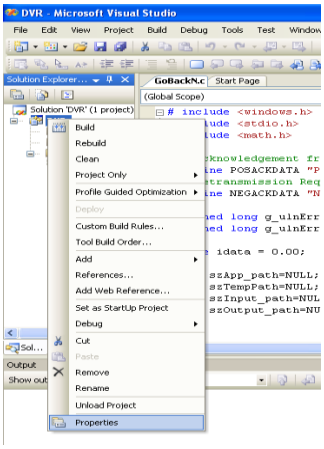
How to De-bug your code linked to NetSim's Programming Exercise

In NetSim, programming exercise menu, users can write and link their code as executables (*.exe). If the user's *.exe file does not provide the correct output, NetSim UI will display the message "Error in User Code". To de-bug your code on getting this message, follow the steps given below:

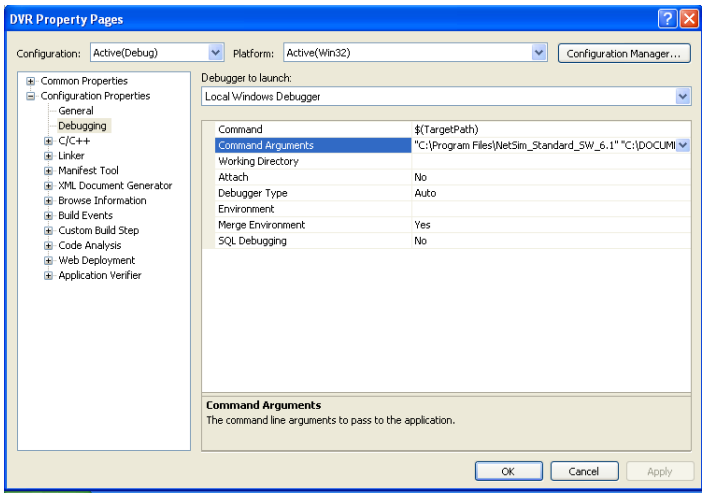
1. Run the scenario again in sample mode. On completion, minimize NetSim and keep it running without closing the programming exercise.
2. Open your code in Visual studio 2010 (or in Eclipse development environment). This is how your code should look when opened in Visual studio as a project (Refer visual studio help on how to create a Win 32 based Exe project)



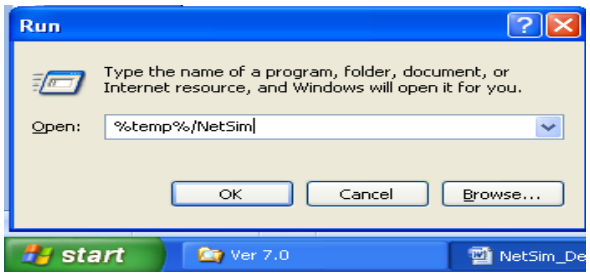
3. Right click on your project and select properties as shown below



4. Inside the properties window select Debugging and edit the Command Arguments as shown



5. Inside command argument add the following two paths
   a. Path to where NetSim is installed within double quotes " ". This is usually C:\Program Files\NetSim Standard. This can be got by right clicking on the NetSim icon and selecting Find target or open file location.
   b. The windows temporary path which has the NetSim folder for temporary data. This can be got by Start->Run and typing %temp%/NetSim



   c. On clicking this, you will get a path similar to C:\Documents and Settings\George\Local Settings\Temp\NetSim. As mentioned above, this should also be in double quotes " ", and there should be a single space between the first path and the second path. For example: "C:\Program Files\NetSim_Standard_SW_6.2" "C:\Documents and Settings\George\Local Settings\Temp\NetSim
6. Now add a breakpoint to your code in your function or in the main ( ), and proceed with debugging
7. At the end check if the output.txt present in the %temp%\NetSim path and the temp.txt present in the %temp%\NetSim path are exactly similar. Exact similarity would indicate that your code will work fine when you run it in use mode the next time.

Address Resolution Protocol

Programming Guidelines

This section guides the user to link his/her own code for Address Resolution Protocol to NetSim.

Pre - Conditions

The user program should read the input from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%" → NetSim → "Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and

The results of the program should be written into the output file **Output.txt**.

Input File Format	Output File Format
The first line of the ***Input.txt*** contains the number of **nodes** in the network, following which the IP Addresses are assigned to each node (ascending order). The last two lines are the Source node IP Address and the Destination node IP Address respectively. An example format of ***input.txt***is given below, Number_of_Nodes=3 Node1_IP_Address=192.168.1.1 Node2_IP_Address=192.168.1.2 Node3_IP_Address=192.168.1.3 Source_Node_IP_ Address=192.168.1.3 Destination_Node_IP_Address=192.168.1.1	The output file should contain two lines, The first line has the details of the Source (i.e.) in which class it is present. The second line has the result of the ARP i.e., whether destination is present in the class. The second line has the flag value (**important)** that is used for the animation. The result for the above is: Source is Class C.Destination is present in that class 2

Interface Source Code

Interface Source code written in C is given and using this the user can write only the Address Resolution Protocol inside the function fnARP () using the variables already declared.

To view the interface source code, go to

NetSim Installation path / src / Programming / ARP.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista
- Properties --> find target in Windows XP.

Sample Scenario:

Objective -To find the Medium Access Control (MAC**) Address of the Destination Node using** Address Resolution Protocol (ARP).

How to Proceed? - The objective can be executed in NetSim by using the programming exercise available. In the **Programming** menu select **Address Resolution Protocol(ARP)**.

Sample Input - By using the **Input Panel** that is available on the left hand side a **Sample Scenario** is created. The Steps involved in creating are as follows,

- **Sample Mode** should be selected.
- **Click** on the drop down button and select 6 **Nodes**.
- List of **NodesNumbers** along with their **IPAddresses** that would be availed are,

Node Numbers	IP Address
1	192.168.1.1
2	192.168.1.2
3	192.168.1.3
4	192.168.1.4
5	192.168.1.5
6	192.168.1.6

- **ARP Request System** and **ARP Reply System** needs to be selected. That is,
  1. **ARP Request System→** 192.168.1. "1" (Any one **Node Number** to be selected)
  2. **ARP Reply System→** 192.168.1. "6" (Any one **Node Number** to be selected)
- Then **Run** button need to be **clicked**.

Output - Output for the above **Sample** is as follows,

- The **Source Node** (i.e. **Node Number** 1 in the above example) sends the **ARP Request** to the **Connecting Device**.
- The **Connecting Device** then broadcasts the **ARP Request** to all the **Nodes** available in the network.
- The **Destination Node** (i.e. **Node Number** 6 in the above example) sends an acknowledgement in the form of **ARP Reply** (i.e. The **Destination MAC - MAC Address** of the **Destination Node**) to the **Connecting Device**.
- The **Device** then transmits the **ARP Reply** (i.e. The **Destination MAC - MAC Address** of the **Destination Node**) only to the **Source Node**.
- Once the sample experiment is done, then **Refresh** button can be clicked to create new samples.

Assignment of Sites to Concentrator

Programming Guidelines

This section guides the user to link his/her own code for Assignment of Sites to Concentrator to NetSim.

Pre - Conditions

The user program should read the inputted scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%" → NetSim → "Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and,

The results of the program should be written into the output file **Output.txt**.

Input file format	**Output file format**
Input.txt contains the number of **sites**, number of **concentrators**, and number of sites that can be assigned to a concentrator, the order in which sites are to be assigned and the distance matrix given by the user. The format of ***input.txt***is: Number_of_Sites=2 Number_of_Concentrators=2 Sites_per_Concentrator=1 Selected_Priority=1>0> Distance 1>2> 3>4> **Note:** '**>'** is the delimiter symbol, which is used to separate each input.	In the output file (i.e. **'Output.txt'**) each line should indicate one complete traverse. For example, for the above input, the output file should be: 2>1> 1>2> In each line, the first character indicates the **site** and the rest indicates the **concentrators. The** second character (excluding the '>' symbol) is the first **concentrator** found with minimum distance/cost. The third character is a **concentrator,** which has the minimum distance/cost compared to the previous one and so on. **Note:**'**>'** is the delimiter symbol, which is used to separate each input.

Interface Source Code

Interface Source code code written in C is given using this the user can write only the Assignments of Sites to Concentrators inside the function fnAssignmentsofSites() using the variables already declared.

To view the interface source code, go to

NetSim Installation path / src / Programming/ AssignSites.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista
- Properties --> find target in Windows XP.

Sample Scenarios:

Objective - To allocate the resource which is concentrator to the require system which is sites based on the distance for each sites to concentrator.

How to proceed? - The objective can be executed in NetSim using the programming exercise available, under programming user has to select Assignment of Sites to Concentrator.

Sample Input - By using the **Input Panel** that is available on the left hand side a **Sample Scenario** is created. The Steps involved in creating are as follows,

- **Sample Mode** should be selected.
- Number of Sites need to be **selected**. The value that has selected ranges from **1** to **7**.
- Number of Concentrators need to be **selected**. The values available for selection are **2**, **3** and **4**.
- Number of Sites / Concentrators need to be **selected**. The value that has selected ranges from **1** to **7**.
- **Click** on the image to **select** Priority. **Click** on Change Priority to reset the Priority of the Sites.
- **Enter** the Distance in the given table. Distance should be in the range, **1** to **99** km.
- Then **Run** button need to be **clicked**. **Refresh** button can be used if new Inputs have to be given.
- **Click** on Concept, Algorithm, Pseudo Code & Flow Chart to get help on it.

Output - The following steps are under gone internally,

- The site which has the highest priority searches for a concentrator which is the nearest.

- A red line appears between the site and a concentrator. This indicates the only shortest path available between the site and the concentrators. Hence, it indicates the site has been allocated to that concentrator.
- A site can have only one concentrator, whereas a concentrator can have many sites linked to it.
- Once the sample experiment is done, then **Refresh** button can be **clicked** to create New Samples.

Cryptography - Substitution - Encryption

Programming Guidelines

This section guides the user to link his/her own code for Substitution to NetSim.

Pre-conditions

The user program should read the inputted scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and, the results of the program should be written into the output file **Output.txt**.

Input File Format	Output File Format
There are six lines in Input.txt file. Cryptographic_Technique=Substitution Cryptographic_Method=Encryption Plain_Text: tetcos Key_Text: 2	Plain letter>encrypted letter for the corresponding plain letter> t>u>v> e>f>g> t>u>v> c>d>e> o>p>q> s>t>u>

Interface Source Code

Interface Source code written in C is given using this the user can write only the Substitution-Encryption inside the function fnSubstutionEncryption () using the variables already declared.

To view the interface source code, go to

NetSim Installation path / src / Programming/ SubstEncrypt.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista
- Properties --> find target in Windows XP.

Cryptography - Substitution - Decryption

Programming Guidelines

This section guides the user to link his/her own code for Substitution to NetSim.

Pre - Conditions

The user program should read the inputted scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%" → NetSim → "Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and,

The results of the program should be written into the output file **Output.txt**.

Input File Format	Output File Format
There are six lines in Input.txt file. Cryptographic_Technique=Substitution Cryptographic_Method=Decryption Cipher_Text: vgvequ Key_Text: 2	Cipher letter>decrypted letter for the corresponding cipher letter> v>u>t> g>f>e> v>u>t> e>d>c> q>p>o> u>t>s>

Interface Source Code

Interface Source code written in C is given using this the user can write only the Substitution-Decryption inside the function SubstutionDecryption () using the variables already declared.

To view the interface source code, go to

NetSim Installation path / src / Programming/ SubstDecrypt.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7

- Open file location in Windows Vista
- Properties --> find target in Windows XP.

Sample Scenarios:

Objective - Encrypt and decrypt the message with the same key value using Substitution.

How to Proceed? - The objective can be executed in NetSim using the programming exercise available. Under programming user has to select **Cryptography → Substitution.**

❖ **Encryption:**

Sample Input: In the Input panel the following steps need to be done,

- **SampleMode** should be **selected**.
- Encryption need to be selected.
- Enter the Plain Text that needs to be encrypted. Maximum of 8 alphabets need to be entered.
- Enter the Key Value. This is an Integer which is within the range **0** to **26**.
- Then **Run** button need to be **clicked**. **Refresh** button can be used if new Inputs have to be given.

Output - The following steps are under gone internally,

- The Plain Text and Key Value entered would be displayed in red color.
- The corresponding Cipher Text would be obtained.
- Once the sample experiment is done, then **Refresh** button can be **clicked** to create New Samples.

❖ **Decryption:**

Sample Input: In the Input panel the following steps need to be done,

- **SampleMode** should be **selected**.
- Decryption need to be selected.

- Cipher Text obtained while encryption is filled in the Plain Text and also the Key Text is same as that entered while encrypting.
- Then **Run** button need to be **clicked**. **Refresh** button can be used if new Inputs have to be given.

Output - The following steps are under gone internally,

- The Cipher Text and Key Value entered would be displayed in red color.
- The corresponding Plain Text that had been entered at the time of encrypting is obtained.
- Once the sample experiment is done, then **Refresh** button can be **clicked** to create New Samples.

Cryptography - Transposition - Encryption

Programming Guidelines

This section guides the user to link his/her own code for Transposition Encryption to NetSim.

Pre-conditions

The user program should read the inputted scenario from text file named '**Input**' with extension txt.The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%" → NetSim → "Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and,

The results of the program should be written into the output file **Output.txt**.

Input File Format	Output File Format
Cryptographic_Technique=Transposition Cryptographic_Method=Encryption Plain_Text: TETCOS Key_Text: BLORE	1>3>4>5>2> T>E>T>C>O>S>T>E>T>C> T>S>O>C>E>T>T>E>C>T>

Interface Source Code

Interface Source code written in C is given using this the user can write only the Transposition-Encryption inside the function fnTranspositionEncryption() using the variables already declared.

To view the interface source code, go to

NetSim Installation path / src / Programming/ TranspEncrypt.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista
- Properties --> find target in Windows XP.

Cryptography - Transposition – Decryption

Programming Guidelines

This section guides the user to link his/her own code for Transposition Decryption to NetSim.

Pre - Conditions

The user program should read the inputted scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%" → NetSim → "Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and,

The results of the program should be written into the output file **Output.txt**.

Input File Format	Output File Format
Cryptographic_Technique=Transposition Cryptographic_Method=Decryption Cipher_Text: TSOCETTECT Key_Text: BLORE	BELOR> 1>5>2>3>4> T>O>E>T>C>S>C>T>E>T>

Interface Source Code

Interface Source code written in C is given using this the user can write only the Transposition-Decryption inside the function fnTranspositionDecryption() using the variables already declared.

To view the interface source code, go to

NetSim Installation path / src / Programming/ TranspDecrypt.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista
- Properties --> find target in Windows XP.

Sample Scenarios:

Objective - Encrypt and decrypt the message with the same key text using Transposition.

How to Proceed? - The objective can be executed in NetSim using the programming exercise available. Under programming user has to select **Cryptography → Transposition**.

❖ Encryption:

Sample Input: In the Input panel the following steps need to be done,

- **SampleMode** should be **selected**.
- Encryption need to be selected.
- Enter the Cipher Text. Maximum of 14 Characters can be entered.
- Enter the Key Value. Maximum of 8 alphabets can be entered.
- Then **Run** button need to be **clicked**. **Refresh** button can be used if new Inputs have to be given.

Output - The following steps are under gone internally,

- The entered Key Value would be displayed first.
- The order of the Key Value is internally sensed.
- The corresponding Cipher Text is obtained.
- Once the sample experiment is done, then **Refresh** button can be **clicked** to create New Samples.

❖ Decryption:

Sample Input: In the Input panel the following steps need to be done,

- Once Encryption is done Decryption has to be selected.
- The Cipher Text obtained for Encryption is the Cipher Text for Decryption. This would be automatically taken. Maximum of 14 Characters can be entered.

- Enter the Key Value will also be taken internally. Maximum of 8 alphabets can be entered.
- Then **Run** button need to be **clicked**. **Refresh** button can be used if new Inputs have to be given.

Output - The following steps are under gone internally,

- The entered Key Value would be displayed first. This is arranged in such a way that the order is changed when compared to Encryption.
- The order of the Key Value is internally sensed.
- The corresponding Plain Text is obtained. This Plain Text would be similar to the Plain Text entered at the time of Encryption.
- Once the sample experiment is done, then **Refresh** button can be **clicked** to create New Samples.

Cryptography - XOR - Encryption

Programming Guidelines

This section guides the user to link his/her own code for XOR Encryption to NetSim.

Pre - Conditions

The user program should read the inputted scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and,

The results of the program should be written into the output file **Output.txt**.

Input File Format	**Output File Format**
Cryptographic_Technique=XOR	01010100>01000010>00010110>
Cryptographic_Method=Encryption	01000101>01001110>00001011>
Plain_Text:	01010100>01001100>00011000>
TETCOS	01000011>01001111>00001100>
Key_Text:	01001111>01010010>00011101>
BNLORE	01010011>01000101>00010110>

Interface Source Code

Interface Source code written in C is given using this the user can write only the XOR-Encryption inside the function fnXOREncryption () using the variables already declared.

To view the interface source code, go to

NetSim Installation path / src / Programming/ XorEncrypt.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista
- Properties --> find target in Windows XP.

Cryptography - XOR - Decryption

Programming Guidelines

This section guides the user to link his/her own code for XOR Decryption to NetSim.

Pre - Conditions

The user program should read the inputted scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type**"%temp%"→NetSim→** "**Input.txt**" and "**Output.txt**"

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and,

The results of the program should be written into the output file **Output.txt**.

Input File Format	Output File Format
Cryptographic_Technique=XOR Cryptographic_Method=Decryption Cipher_Text: 0001011000001011000110000000110000011110100010110 Key_Text: BNLORE	00010110>01000010>01010100> 00001011>01001110>01000101> 00011000>01001100>01010100> 00001100>01001111>01000011> 00011101>01010010>01001111> 00010110>01000101>01010011>

Interface Source Code

Here a skeleton code written in C is given using this the user can write only the XOR-Decryption inside the function fnXORDecryption () using the variables already declared.

To view the interface source code, go to

NetSim Installation path / src / Programming/ XorDecrypt.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista
- Properties --> find target in Windows XP.

Sample Scenarios:

Objective - Encrypt and **Decrypt** the message by using the same **KeyText** in **XOR**.

How to Proceed? - The objective can be executed in **NetSim** using the programming exercise available. In the **Programming** menu select **Cryptography →XOR**.

Note:Encryption should be done first and then **Decryption** should be done.

❖ **Encryption:**

Sample Inputs - By using the **InputPanel** that is available on the left hand side a **SampleScenario** is created. The Steps involved in creating are as follows,

- **SampleMode** should be selected.
- **EncryptionMode** needs to be selected.
- **PlainText** and **KeyValue** need to be entered in the fields available. That is,
  - **Plain Text→** tetcos (Maximum of 8 Characters)
  - **Key Text →** 123456 (Maximum of 8 Characters)

  Note: If the length of the **PlainText** and **Key Value** differs then an error would pop out.
- Then **Run** button need to be clicked.

Output - Output for the above Sample is as follows,

- **Plain Text** → t e t c o s
- **Key Text** → 1 2 3 4 5 6
- **Binary of plain text** → 01110011
- **Binary of key text** → 00110110
- **XOR Value** → 01000101
- **ASCII Equivalent** → 69 87 71 87 90 69
- **Cipher Text** →E W G W Z E

❖ Decryption:

Sample Inputs - By using the **InputPanel** that is available on the left hand side a **Sample Scenario** is created. The Steps involved in creating are as follows,

- **SampleMode** should be selected.
- After completing the **Encryption** part, **DecryptionMode** needs to be selected.
- **CipherText** (this field is automatically filled) and **KeyValue** (this should be same as the one that is entered in Encryption) need to be entered in the fields available. That is,
  - **CipherText→** 01000101010101110100011101010111010110100100 0101 (Cipher Text in Binary Format Maximum 64 bits)
  - **KeyText→** 123456 (Maximum of 8 Characters)

  Note - Both **Cipher Text** and **Key Text** is automatically taken when **Decryption** button is selected (i.e. after undergoing **Encryption**).
- Then **Run** button need to be clicked.

Output - Output for the above **Sample** is as follows,

- **Cipher Text** →E W G W Z E
- **Key Text** → 1 2 3 4 5 6
- **Binary of cipher text** → 01000101
- **Binary of key text** → 00110110
- **XOR Value** → 01110011
- **ASCII Equivalent** → 116 101 116 99 111 115
- **Plain Text** → t e t c o s

Note -

- **Text** in the **Input** of the **Encryption** and **Output** of the **Decryption** should be the same if the **Key Value** is same.
- The **Cipher Text** in case of **Encryption** is **Alpha Numeric** (i.e. for the user to understand in a better manner)and in case of **Decryption** the tool converts it to **Binary form** (i.e. since, the tool doesn't recognize **Alpha Numeric**).

Cryptography - Data Encryption Standard (DES) - Encryption

Programming Guidelines

This section guides the user to link his/her own code for DES Encryption to NetSim.

Pre - Conditions

The user program should read the inputted scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

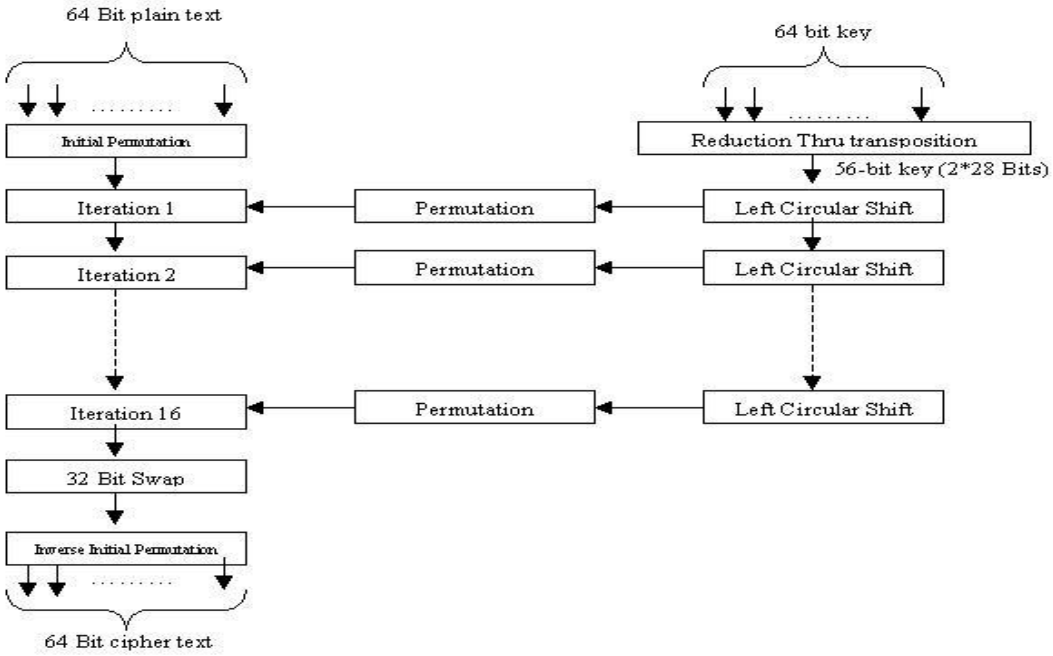
Note: The temporary directory is navigated through the following step.

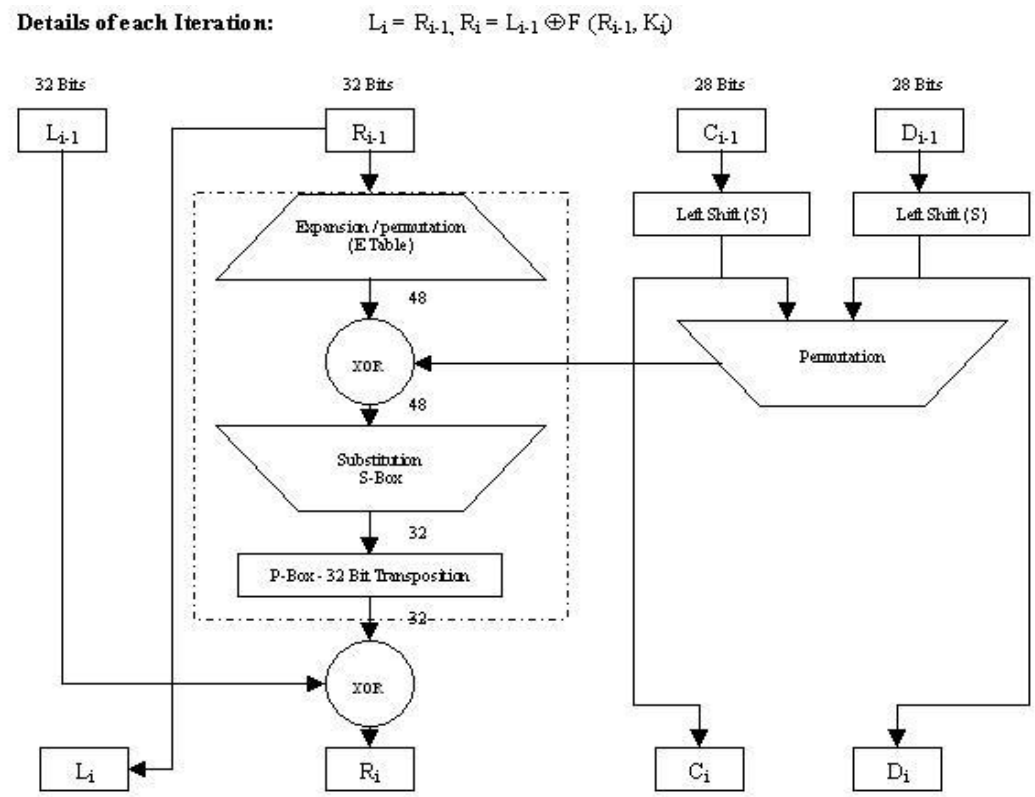
Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

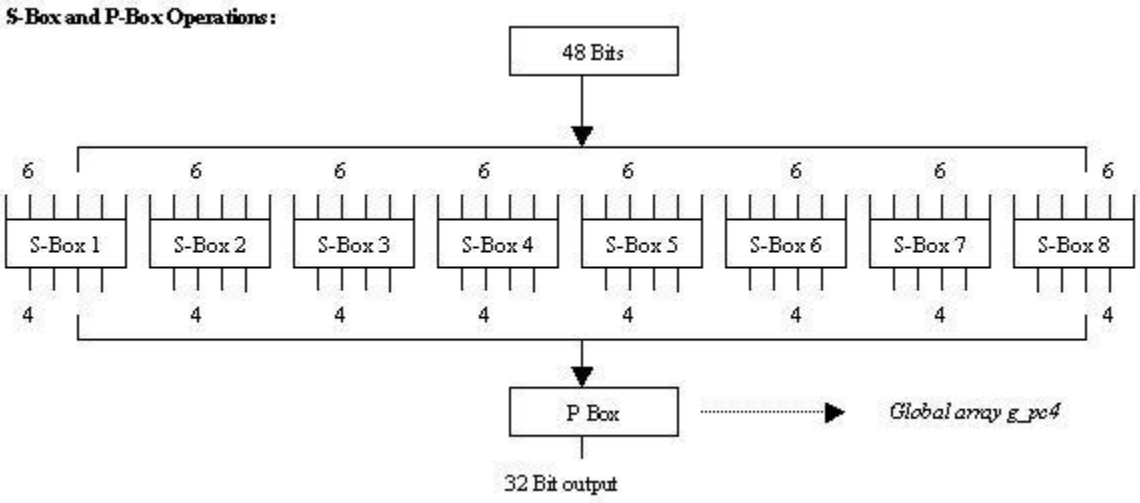
General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and,The results of the program should be written into the output file **Output.txt**.







6bit - 4 Bit conversion logic (in each S-Box):

Of the 6 input bits, 4 bits (1, 2, 3, and 4) are used for identifying the S-Box column (4 bits representing 16 columns) and the remaining 2 bits (0, 5) are used for identifying the S-Box row (2 bits representing 4 rows). The corresponding hexadecimal number is chosen from the S-box. Since each Hexadecimal number represents 4 bits in binary form, the total output is of 32 bits.

Input File Format	Output File Format
Cryptographic_Method=Encryption Key_Text=abcdef1234567890 No_of_Iterations=1 Data=tetcos	The number of lines present depends on the number of iterations chosen. The number of lines equals the number of iterations, plus one. The last line of inputs gives the data (encrypted data if encryption has been chosen, else the decrypted data if decryption has been chosen). The previous lines give the DES Key generated for encryption or decryption. 1101010101111100100110101100001011100110000110016160757 33F724B40

Interface Source Code

Interface Source code written in C is given using this the user can write only the DES-Encryption inside the function fnDESEncryption() using the variables already declared.

To view the interface source code, go to

NetSim Installation path / src / Programming/ DesEncrypt.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista
- Properties --> find target in Windows XP.

Cryptography - Data Encryption Standard (DES) - Decryption

Programming Guidelines

This section guides the user to link his/her own code for DES Decryption to NetSim.

Pre - Conditions

The user program should read the inputted scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

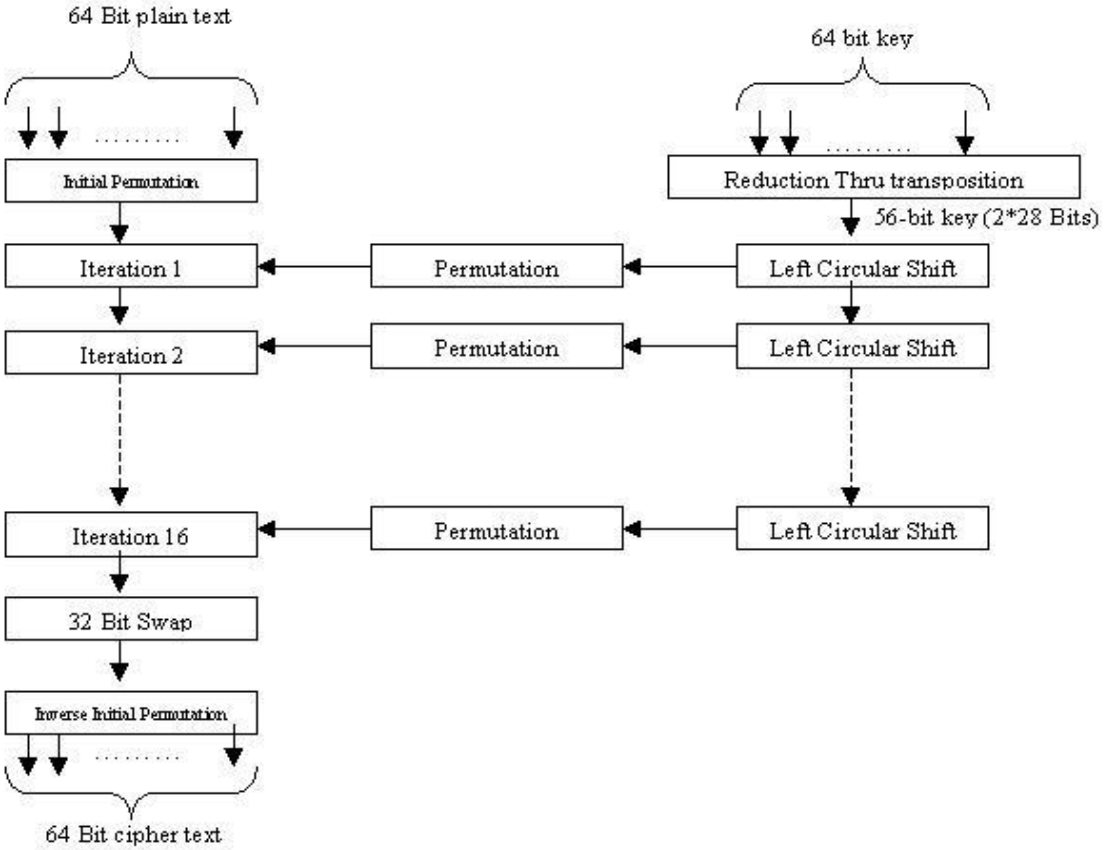
Run →Type **"%temp%" → NetSim → "Input.txt"** and **"Output.txt"**

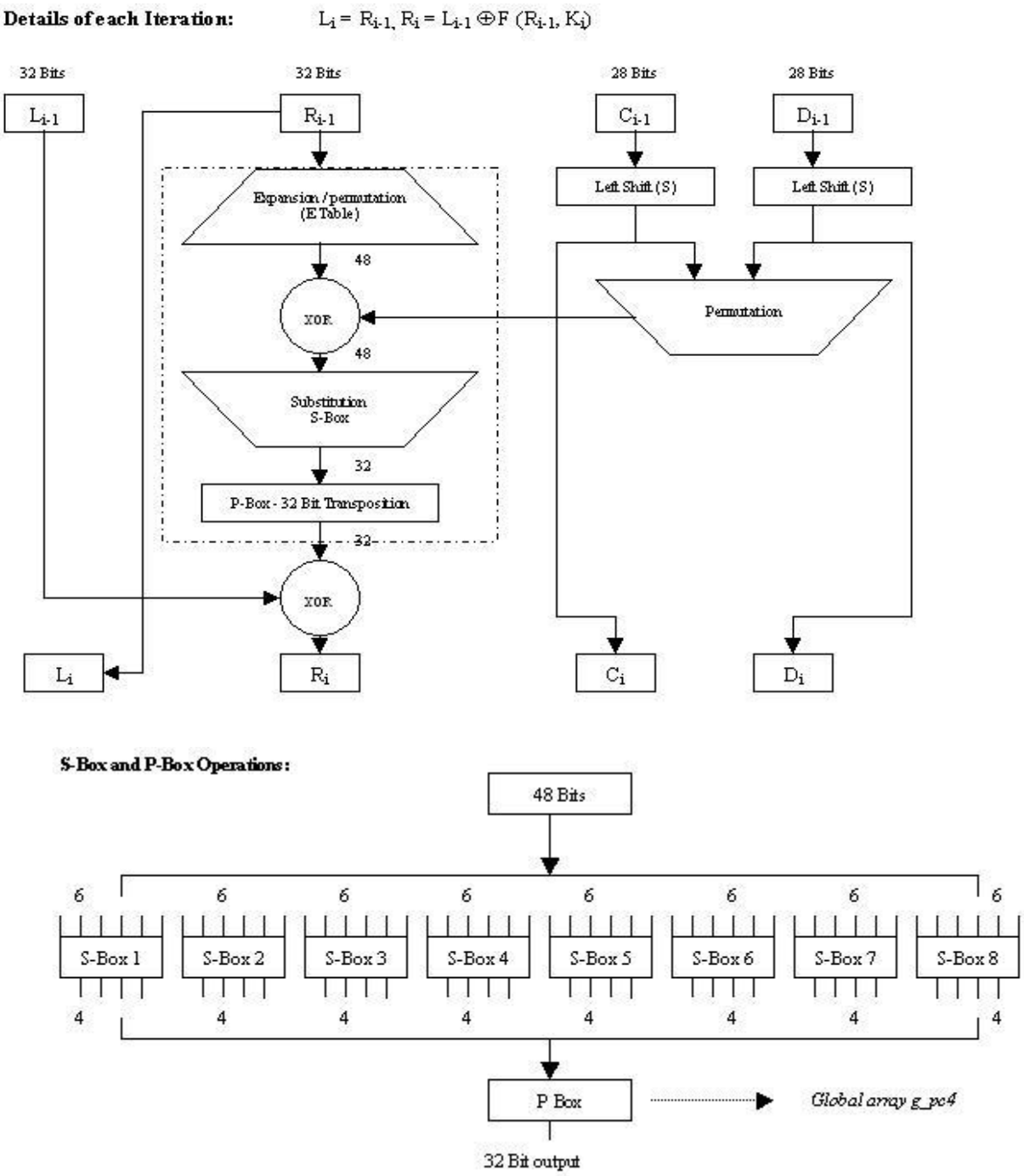
General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and,

The results of the program should be written into the output file **Output.txt**.





6bit - 4 Bit conversion logic (in each S-Box):

Of the 6 input bits, 4 bits (1, 2, 3, 4) are used for identifying the S-Box column (4 bits representing 16 columns) and the remaining 2 bits (0, 5) are used for identifying the S-

Box row (2 bits representing 4 rows). The corresponding hexadecimal number is chosen from the

S-box. Since each Hexadecimal number represents 4 bits in binary form, the total output is of 32 bits.

Input File Format	Output File Format
Cryptographic_Method=Decryption Key_Text=abcdef1234567890 No_of_Iterations=1 Data=616075733F724B40	The number of lines present depends on the number of iterations chosen. The number of lines equals the number of iterations, plus one. The last line of inputs gives the data (encrypted data if encryption has been chosen, else the decrypted data if decryption has been chosen). The previous lines give the DES Key generated for encryption or decryption. 11010101011111001001101011000010111001100001100 1 tetcos

Interface Source Code

Interface Source code written in C is given using this the user can write only the DES-Decryption inside the function fnDESDecryption () using the variables already declared.

To view the interface source code, go to

NetSim Installation path / src / Programming/ DesDecrypt.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista
- Properties --> find target in Windows XP.

Sample Scenarios:

Objective - Encrypt and decrypt the message with the using DES.

How to Proceed? - The objective can be executed in NetSim using the programming exercise available. Under programming user has to select **Cryptography→Advanced→DataEncryptionStandard**.

1. Encryption:

Sample Inputs - In the Input panel the following steps need to be done,

- **SampleMode** should be **selected**.
- Encryption need to be **selected**.
- Enter the Key Text in the provided field. Only hexadecimal characters have to be entered. Maximum of 16 characters can be entered in this field. Characters more than 16 will be filled in the Data field.
- Number of Iterations need to be selected. The value ranges from **1** to **16**.
- Data needs to be entered in this field that has to be encrypted. Data can be entered only when the Enter Key Text field is filed with 16 hexadecimal characters. Maximum of 1500 characters can be stuffed into this field.
- Then **Run** button need to be **clicked**. **Refresh** button can be used if new Inputs have to be given.

Output -

- Key1 (i.e. if Iteration is 1) in binary format is obtained.
- The corresponding Cipher text is obtained.
- Click on Copy button and then on the Paste button to make use of the Cipher text for Decryption.
- Once the sample experiment is done, then **Refresh** button can be **clicked** to create New Samples.

2. Decryption:

Sample Inputs - In the Input panel the following steps need to be done,

- **SampleMode** should be **selected**.

- Once the Encryption is done, click on Copy and Paste buttons. The encrypted data is filled in the data field available in the Decryption view.
- Fields such as Enter the Key Text, Number of Iteration and Data is filled in automatically when Copy and Paste button is clicked.
- Then **Run** button need to be **clicked**. **Refresh** button can be used if new Inputs have to be given.

Output - The following steps are under gone internally,

- Key1 (i.e. if Iteration is 1) in binary format is obtained.
- The Data that was entered in the Encryption view would be encrypted and displayed.
- Once the sample experiment is done, then **Refresh** button can be **clicked** to create New Samples.

Rivest-Shamir - Adleman Algorithm (RSA)

Programming Guidelines

This section guides the user to link his/her own code for RSAAlgorithmto NetSim.

Pre - Conditions

The user program should read the inputted scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

The path of the input file and the output file can be viewed on clicking the Button "Path" in NetSim.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%" → NetSim → "Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and,

The results of the program should be written into the output file **Output.txt**.

Input file format	Output file format
Input.txt file has the input data. The format is: Plain_Text=T	**dt1>dt2>dt3>dt4>dt5>dt6>dt7>dt8>** dt 1: specifies the prime P dt 2: specifies the prime Q dt 3: specifies the value of N dt 4: specifies the value of Z dt 5: specifies the value of Kp dt 6: specifies the value of Ks dt 7: specifies the value of the Cipher Text dt 8: specifies the value of the Plain Text Example: For input: 'T' Output: 11>3>33>20>7>3>14>20>

Interface Source Code

To view the interface source code, go to

NetSim Installation path / src / Programming/ RSA.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista
- Properties --> find target in Windows XP.

Sample Scenario:

Objective - Encrypt and decrypt the message with the using DES.

How to Proceed? - The objective can be executed in NetSim using the programming exercise available. Under programming user has to select **Cryptography→Advanced→RSA**.

Sample Inputs - In the Input panel the following steps need to be done,

- **SampleMode** should be **selected**.
- Plain Text of only one character has to be entered in the field available.
- Then **Run** button need to be **clicked**. **Refresh** button can be used if new Inputs have to be given.

Output - The following steps are under gone internally,

- Value of the prime numbers P and Q are obtained.
- Value of N[p*q] is obtained.
- Value of Z [(p-1)*(q-1)] is obtained.
- Value of Kp and Ks is obtained
- Plain Text which is the actual count of the alphabet is obtained.
- The corresponding Cipher Text is obtained.
- Once the sample experiment is done, then **Refresh** button can be **clicked** to create New Samples.

Cryptography - Wired Equivalent Privacy (WEP) – Encryption

Programming Guidelines

This section guides the user to link his/her own code for WEP Encryption to NetSim.

Pre - Conditions

The user program should read the inputted scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%" → NetSim → "Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and,

The results of the program should be written into the output file **Output.txt**.

Input File Format	Output File Format
Cryptographic_Method=Encryption Data: TETCOS	There are 3 lines in output.txt file Initialization Vector Key Text Cipher Text **Ex:** 8A699C 0123456789 364130B9CA018D5B760383CD85528751

Interface Source Code

Interface Source code written in C is given using this the user can write only the WEP-Encryption inside the function fnWEPEncrypt () using the variables already declared.

To view the interface source code, go to

NetSim Installation path / src / Programming/ WEPEncrypt.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista
- Properties --> find target in Windows XP.

Cryptography - Wired Equivalent Privacy (WEP) - Decryption

Programming Guidelines

This section guides the user to link his/her own code for WEP Decryption to NetSim.

Pre - Conditions

The user program should read the inputted scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note:

The temporary directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and,

The results of the program should be written into the output file **Output.txt**.

Input File Format	Output File Format
Cryptographic_Method=Decryption Initialization_Vector=8A699C Key=0123456789 Data: 364130B9CA018D5B760383CD85528751	Plain text **Ex:** TETCOS

Interface Source Code

Interface Source code written in C is given using this the user can write only the WEP-Decryption inside the function fnWEPDecrypt () using the variables already declared.

To view the interface source code, go to

NetSim Installation path / src / Programming/ WEPDecrypt.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista
- Properties --> find target in Windows XP.

Sample Scenarios:

Objective - Encrypt and **Decrypt** the message by using **WEP**.

How to Proceed? - The objective can be executed in **NetSim** using the programming exercise available. In **Programming** menu select **Cryptography→Advanced→Wired Equivalent Privacy**.

Note: **Encryption** should be done first and then **Decryption** should be done.

❖ **Encryption**

Sample Inputs - By using the **Input Panel** that is available on the left hand side a **SampleScenario** is created. The Steps involved in creating are as follows,

- **SampleMode** should be selected.
- Under **CryptographicMethod**, **Encryption Mode** needs to be selected.
- **Input→Data** (Maximum of 1500 Characters) - **NetSim TetCos Bangalore**
- Then **Run** button need to be clicked.

Sample Output - Output for the above **Sample** is as follows,

- **Initialization Vector** : 3E41C6
- **Key** : ABCDEF0123

- **The Encrypted Text obtained -** "786441A955C419F9537576954F897BCC5866549653DB5CA73E1B4DF63CA939AD"
- **Click** on **Copy** button and then on **Paste** button. This would **Copy** the **EncryptedText** onto the **DecryptionInputDatafield**.

❖ Decryption

Sample Inputs - By using the **InputPanel** that is available on the left hand side a **Sample Scenario** is created. The Steps involved in creating are as follows,

- **Sample Mode** should be selected. (This is **Automatically** selected once the **Paste** button is clicked at the time of **Encrypting** the **Data**).
- Under **Cryptographic Method**, **Encryption Mode** needs to be selected. (This is **Automatically** selected once the **Paste** button is clicked at the time of **Encrypting** the **Data**).
- **Enter Key Value→** 3E41C6ABCDEF0123 (16Hexadecimal Characters). (This field is **Automatically** filled once the **Paste** button is clicked at the time of **Encrypting** the **Data**).
- **Number of Iterations→** 1.
- **Input→Data -** "786441A955C419F9537576954F897BCC5866549653DB5CA73E1B4DF63CA939AD" (This field is **Automatically** filled once the **Paste** button is clicked at the time of **Encrypting** the **Data**).
- Then **Run** button need to be clicked.

Sample Output - Output for the above **Sample** is as follows,

- **Initialization Vector** : 3E41C6
- **Key** : ABCDEF0123
- The **EncryptedText** obtained - **NetSim TetCos Bangalore**

Distance Vector Routing

Programming Guidelines

This section guides the user to link his/her own code for Distance Vector Routing to NetSim.

Pre - Conditions

The user program should read the input from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%" → NetSim → "Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and

The results of the program should be written into the output file **Output.txt**.

Input File Format	Output File Format
Router_ID=1>Router_Name=Router_1>No_Of_Neighbour=2>Neighbours_ID=2>5 Router_ID=2>Router_Name=Router_2>No_Of_Neighbour=1>Neighbours_ID=1 Router_ID=3>Router_Name=Router_3>No_Of_Neighbour=0 Router_ID=4>Router_Name=Router_4>No_Of_Neighbour=0 Router_ID=5>Router_Name=Router_5>No_Of_Neighbour=1>Neighbours_ID=1 Router_ID=6>Router_Name=Router_6>No_Of_Neighbour=0	Initial Stage>Source Router ID - 1>Destination RouterID>CostID>Intermediate RouterID Ex: 0>1>2>1>0 0>1>5>1>0 0>2>1>1>0 0>5>1>1>0 1>1>2>1>0 1>1>5>1>0

	1>2>1>1>0 1>2>5>2>1 1>5>1>1>0 1>5>2>2>1 2>1>2>1>0 2>1>5>1>0 2>2>1>1>0 2>2>5>2>1 2>5>1>1>0 2>5>2>2>1

Interface Source Code

Interface Source code written in C is given using this the user can write only the Distance Vector Routing inside the function fnDistVectAlgorithm() using the variables already declared.

To view the interface source code, go to

NetSim Installation path / src / Programming/ DVR.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista
- Properties --> find target in Windows XP.

Sample Scenarios:

Objective - Find the Shortest Path using Distance Vector Routing.

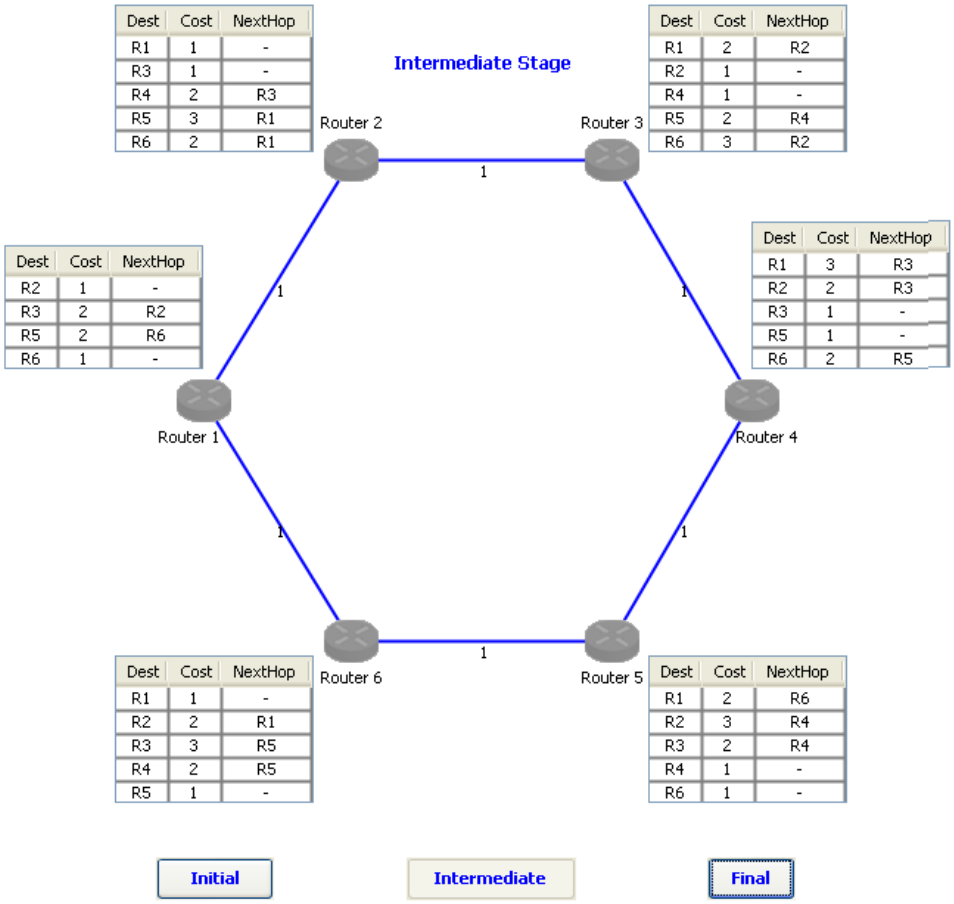
How to Proceed? - The objective can be executed in NetSim by using the programming exercise available. In the **Programming** menu select**Distance Vector Routing.**

Sample Input -Follow the below given steps,

- Click on two Routers to establish a Path (i.e. 1st Click on Router number 1 and 2, then similarly on Router number 2 and 3, 3 and 4, 4 and 5, 5 and 6, & 6 and 1).

- Router 1, 2, 3, 4, 5, 6 are connected.
- Click on **Initial, Intermediate** and **Final** button to execute.

Output - The Output that is obtained is given below,



Initial button can be clicked to view the initial table of the all router.

Intermediate button can be clicked to view the intermediate table of the all router.

Final button can be clicked to view the final table of the all router.

Distance Host Configuration Protocol

Programming Guidelines

This section guides the user to link his/her own code for Dynamic Host Configuration Protocol to NetSim.

Pre - Conditions

The user program should read the input from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%" → NetSim → "Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and

The results of the program should be written into the output file **Output.txt**.

Input File Format	Output File Format
Number_of_Clients = 4 Start_Address = 192.168.0.105 End_Address = 192.168.0.109 Mask = 255.255.255.0	DHCP_DISCOVER> Node1>Node2> Node1>Node3> Node1>Node4> Node1>Server DHCP_OFFER> Server>Node1 DHCP_REQUEST> Node1>Server DHCP_ACK> Server>192.168.0.105 DHCP_DISCOVER> Node2>Node1> Node2>Node3> Node2>Node4> Node2>Server DHCP_OFFER> Server>Node2 DHCP_REQUEST> Node2>Server DHCP_ACK> Server>192.168.0.106

	DHCP_DISCOVER> Node3>Node1> Node3>Node2> Node3>Node4> Node3>Server DHCP_OFFER> Server>Node3 DHCP_REQUEST> Node3>Server DHCP_ACK> Server>192.168.0.107 DHCP_DISCOVER> Node4>Node1> Node4>Node2> Node4>Node3> Node4>Server DHCP_OFFER> Server>Node4 DHCP_REQUEST> Node4>Server DHCP_ACK> Server>192.168.0.108

Interface Source Code

Interface Source code written in C is given using this the user can write only the Dynamic Host Configuration Protocol inside the function fnDHCPServer() and fnDHCPClient() using the variables already declared.

To view the interface source code, go to

NetSim Installation path / src / Programming / DHCP.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista
- Properties --> find target in Windows XP.

Sample Scenarios:

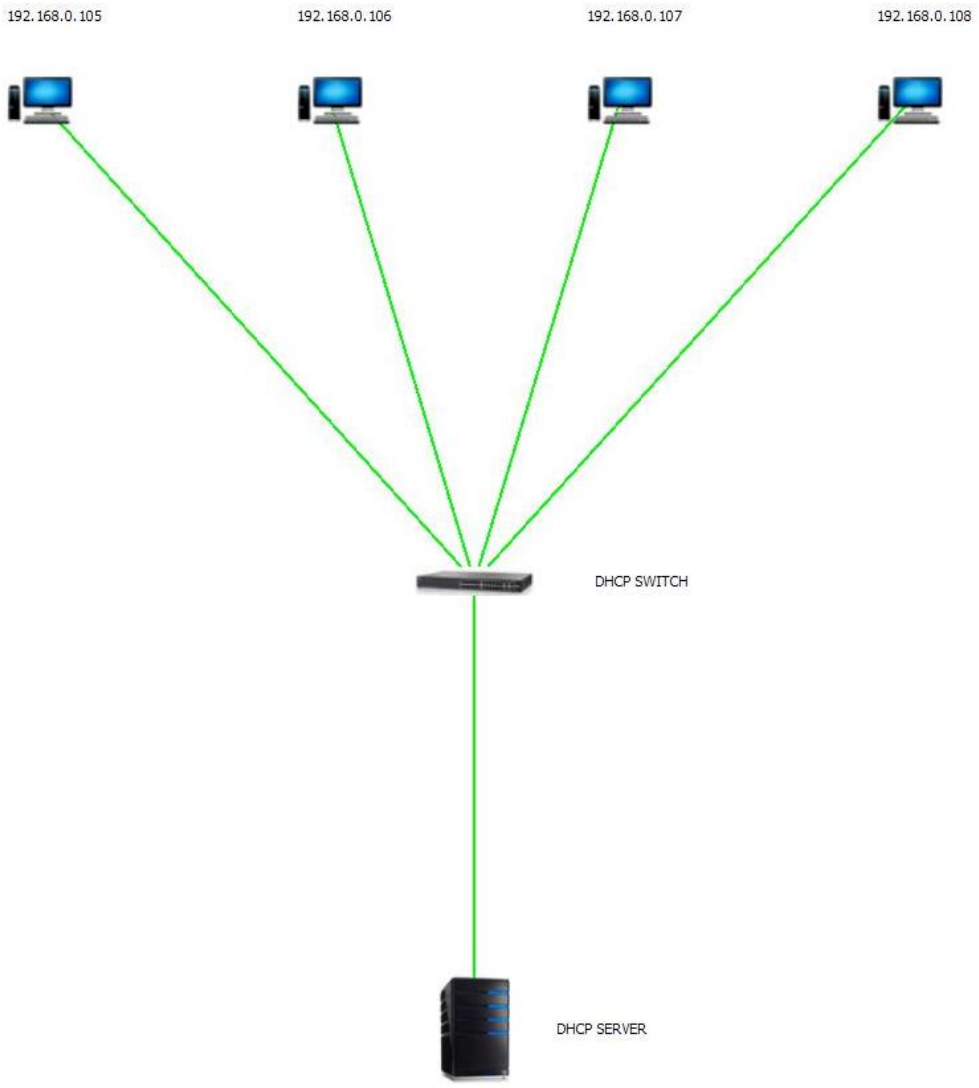
Objective - Find the Shortest Path using Dynamic Host Configuration Protocol.

How to Proceed? - The objective can be executed in NetSim by using the programming exercise available. In the **Programming** menu select**Dynamic Host Configuration Protocol.**

Sample Input -Follow the below given steps,

- Select number of clients.
- Set the Start Address and the End Address
- Click on **Run** button to execute.

Output - The Output that is obtained is given below,



Error Correcting Code - Hamming Code

Programming Guidelines

This section guides the user to link his/her own code for Hamming Code Generator to NetSim.

Pre - Conditions

The user program should read the inputted scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

The path of the input file and the output file can be viewed on clicking the Button "Path" in NetSim.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and,

The results of the program should be written into the output file **Output.txt**.

Input file format	Output file format
Parity=Odd Data=a Error_Position=7 Data_Bits_Original=01100001 Data_Bits_Error=00100001	'**Output.txt** ' file, the first line is the count of check bits to be introduced in the current hamming code application. The next 8 lines is the check bit calculation process, the first four lines are for input data and the next four lines are for the error data. 1 and 2 are the check bits that vary in the Hamming String. The last value is the actual position in the Hamming Code generated where the data bits has been changed. 4 - The number of check bits introduced 3>5>7>9>11>13>0>0>- the check bit of position **1** of input data 3>6>7>10>11>1>1>- the check bit of position **2** of input data

	5>6>7>12>13>0>0>- the check bit of position **4** of input data 9>10>11>12>13>0>0>- the check bit of position **8** of input data 3>5>7>9>11>13>0>0>- the check bit of position **1** of error data 3>6>7>10>11>1>1>- the check bit of position **2** of error data 5>6>7>12>13>1>1>- the check bit of position **4** of error data 9>10>11>12>13>1>1>- the check bit of position **8** of error data 4>8>12>- the check bit position whose value has been changed and the final error position.

Interface Source Code

To view the interface source code, go to

NetSim Installation path / src / Programming/ HammingCode.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista
- Properties --> find target in Windows XP.

Sample Scenarios:

Objective - To detect and correct the single bit error occurs in the transmission of data.

How to proceed? - The objective can be executed in NetSim using the programming exercise available, under programming user has to select **Error Correcting Code → Hamming Code**

Sample Inputs - In the Input panel the following steps need to be done,

- **SampleMode** should be **selected**.
- Parity need to be selected. Either Odd or Even can be selected.
- Data need to be entered. Maximum of 8 alphabets can be entered.
- Error Position needs to be selected. Based on the input, its values ranges.
- Then **Run** button need to be **clicked**. **Refresh** button can be used if new Inputs have to be given.

Output - The following steps are under gone internally,

- Data Bits which if in the binary form is obtained. This Data Bits is obtained for Original Data entered as well as for the Error selected.
- Depending on the Parity selected, tables of Hamming String, Original Data and Error Data will be obtained.
- For the Error Data table to be obtained click on the Next button available in the output panel. When the Error Data table is obtained Error Position value is also obtained.
- Once the sample experiment is done, then **Refresh** button can be **clicked** to create New Samples.

Error Detection Code - Cyclic Redundancy Check (CRC) - 12

Programming Guidelines

This section guides the user to Run his/her own code for Cyclic Redundancy Check to NetSim.

Pre - Conditions

The user program should read the input scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and

The results of the program should be written into the output file **Output.txt**.

Input File Format	**Output File format**
Algorithm=CRC_12 Condition=No_Error File_Path=C:\Users\P.Sathishkumar\Documents\1 Th.txt>	Output contains two values, which is the written in the separate line. The First line has the CRC value of the data (Sender side CRC value). The Second line has the CRC value of the data (Receiver side CRC value). Example: 8CB 000

Interface Source Code

Interface Source code written in C is given using this the user can write only the Cyclic Redundancy Check inside the function fnCRC () using the variables already declared.

To view the interface source code, go to

NetSim Installation path / src / Programming/ Crc12.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista
- Properties --> find target in Windows XP.

Sample Scenarios:

Objective - To detect the error found in the file transferred between a **Sender** and **Receiver** using **CRC12.**

How to Proceed? - The objective can be executed in **NetSim** using the programming exercise available. In the **Programming Menu** select**Error Detecting Codes→Cyclic Redundancy Check**.

Sample Input

- **For No Error Case -**Follow the below given steps,

0. **SampleMode** should be selected.
1. **SelectCRC12** as **Algorithm** from the list available.
2. Under **Condition**, "**NoError**" should be selected.
3. Under **Input**, **Enter** the path of the file name to get its **CRC**. The file should be in "**.txt**" format which should not exceed 5000bytes.
4. **Click** on **Run** button to execute. **Refresh** button can be used if new Inputs have to be given.

- **For Error Case -** Follow the below given steps,
    0. **SampleMode** should be selected.
    1. **SelectCRC12** as **Algorithm** from the list available.
    2. Under **Condition**, "**Error**" should be selected.
    3. Under **Input**, **Enter** the path of the file name to get its **CRC**.The file should be in "**.txt**" format which should not exceed 5000bytes.
    4. **Click** on **Run** button to execute. **Refresh** button can be used if new Inputs have to be given.

Sample Output

- **For No Error Case:** The **CalculatedCRC** should be **Zero** when the "**.txt file**" is received by the **Node2**. The message "**Data Frame** is **Flowing** from **Node1** to **Node2** with **No Error**".
- **For Error Case:** The **CalculatedCRC** should be **Non-Zero** when the **".txt file**" is received by the **Node2**. The message "**Data Frame** is **Flowing** from **Node1** to **Node2** with **Error**".

Error Detection Code - Cyclic Redundancy Check (CRC) – 16

Programming Guidelines

This section guides the user to link his/her own code for Cyclic Redundancy Check to NetSim.

Pre - Conditions

The user program should read the input from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%" → NetSim → "Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and

The results of the program should be written into the output file **Output.txt**.

Input File Format	Output File format
Algorithm=CRC_16 Condition=No_Error File_Path=C:\Users\P.Sathishkumar\Documents\1 Th.txt>	Output contains two values, which is the written in the separate line. The First line has the CRC value of the data (Sender side CRC value). The Second line has the CRC value of the data (Receiver side CRC value). Example: 0FCF 0000

Interface Source Code

Interface Source code written in C is given using this the user can write only the Cyclic Redundancy Check inside the function fnCRC () using the variables already declared.

To view the interface source code, go to

NetSim Installation path / src / Programming/ Crc16.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista
- Properties --> find target in Windows XP.

Sample Scenarios:

Objective - To detect the error found in the file transferred between a **Sender** and **Receiver** using **CRC16.**

How to Proceed? - The objective can be executed in **NetSim** using the programming exercise available. In the **Programming** menu select**Error Detecting Codes→Cyclic Redundancy Check**.

Sample Input

- **For No Error Case -**Follow the below given steps,
    5. **SampleMode** should be selected.
    6. **SelectCRC16** as **Algorithm** from the list available.
    7. Under **Condition**, "**No Error**" should be selected.
    8. Under **Input**, **Enter** the path of the file name to get its **CRC**. The file should be in "**.txt**" format which should not exceed 5000bytes.
    9. **Click** on **Run** button to execute. **Refresh** button can be used if new Inputs have to be given.
- **For Error Case -** Follow the below given steps,
    10. **SampleMode** should be selected.
    11. **SelectCRC16** as **Algorithm** from the list available.
    12. Under **Condition**, "**Error**" should be selected.
    13. Under **Input**, **Enter** the path of the file name to get its **CRC**. The file should be in "**.txt**" format which should not exceed 5000bytes.
    14. **Click** on **Run** button to execute. **Refresh** button can be used if new Inputs have to be given.

Sample Output

- **For No Error Case:** The **CalculatedCRC** should be **Zero** when the "**.txtfile**" is received by the **Node2**. The message "**DataFrame** is **Flowing** from **Node1** to **Node2** with **NoError**".
- **For Error Case:** The **CalculatedCRC** should be **Non-Zero** when the "**.txtfile**" is received by the **Node2**. The message "**DataFrame** is **Flowing** from **Node1** to **Node2** with **Error**".

Error Detection Code - Cyclic Redundancy Check (CRC) - 32

Programming Guidelines

This section guides the user to link his/her own code for Cyclic Redundancy Check to NetSim.

Pre - Conditions

The user program should read the input from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and

The results of the program should be written into the output file **Output.txt**.

Input File Format	Output File format
Algorithm=CRC_32 Condition=No_Error File_Path=C:\Users\P.Sathishkumar\Documents\1Th.txt>	Output contains two values, which is the written in the separate line. The First line has the CRC value of the data (Sender side CRC value). The Second line has the CRC value of the data (Receiver side CRC value). Example: DD8F598B 00000000

Interface Source Code

Interface Source code written in C is given using this the user can write only the Cyclic Redundancy Check inside the function fnCRC () using the variables already declared.

To view the interface source code, go to

NetSim Installation path / src / Programming/ Crc32.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista
- Properties --> find target in Windows XP.

Sample Scenarios:

Objective - To detect the error found in the file transferred between a **Sender** and **Receiver** using **CRC32.**

How to Proceed? - The objective can be executed in **NetSim** using the programming exercise available. In the **Programming** menu select**Error Detecting Codes→Cyclic Redundancy Check**.

Sample Input

- **For No Error Case -**Follow the below given steps,
    0. **SampleMode** should be selected.
    1. **SelectCRC32** as **Algorithm** from the list available.
    2. Under **Condition**, "**NoError**" should be selected.
    3. Under **Input**, **Enter** the path of the file name to get its **CRC**. The file should be in "**.txt**" format which should not exceed 5000bytes.
    4. **Click** on **Run** button to execute. **Refresh** button can be used if new Inputs have to be given.
- **For Error Case -** Follow the below given steps,
    0. **SampleMode** should be selected.
    1. **SelectCRC32** as **Algorithm** from the list available.
    2. Under **Condition**, "**Error**" should be selected.
    3. Under **Input**, **Enter** the path of the file name to get its **CRC**. The file should be in "**.txt**" format which should not exceed 5000bytes.
    4. **Click** on **Run** button to execute. **Refresh** button can be used if new Inputs have to be given.

Sample Output

- **For No Error Case:** The **CalculatedCRC** should be **Zero** when the "**.txtfile**" is received by the **Node2**. The message "**DataFrame** is **Flowing** from **Node1** to **Node2** with **NoError**".
- **For Error Case:** The **CalculatedCRC** should be **Non-Zero** when the "**.txtfile**" is received by the **Node2**. The message "**DataFrame** is **Flowing** from **Node1** to **Node2** with **Error**".

Error Detection Code - Cyclic Redundancy Check (CRC) – CCITT

Programming Guidelines

This section guides the user to link his/her own code for Cyclic Redundancy Check to NetSim.

Pre - Conditions

The user program should read the input scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and

The results of the program should be written into the output file **Output.txt**.

Input File Format	Output File format
Algorithm=CRC_CCITT Condition=No_Error	Output contains two values, which is the written in the separate line.

File_Path=C:\Users\P.Sathishkumar\Documents\1 Th.txt>	The First line has the CRC value of the data (Sender side CRC value). The Second line has the CRC value of the data (Receiver side CRC value). Example: 92BF 0000

Interface Source Code

Interface Source code written in C is given using this the user can write only the Cyclic Redundancy Check inside the function fnCRC () using the variables already declared.

To view the interface source code, go to

NetSim Installation path / src / Programming/ CrcCcitt.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista
- Properties --> find target in Windows XP.

Sample Scenarios:

Objective - To detect the error found in the file transferred between a **Sender** and **Receiver** using **CRC CCITT.**

How to Proceed? - The objective can be executed in **NetSim** using the programming exercise available. In the **Programming** menu select**Error Detecting Codes→Cyclic Redundancy Check**.

Sample Input

- **For No Error Case -**Follow the below given steps,
    0. **SampleMode** should be selected.
    1. **SelectCRCCCITT** as **Algorithm** from the list available.
    2. Under **Condition**, "**NoError**" should be selected.

3. Under **Input**, **Enter** the path of the file name to get its **CRC**. The file should be in "**.txt**" format which should not exceed 5000bytes.
4. **Click** on **Run** button to execute. **Refresh** button can be used if new Inputs have to be given.

- **For Error Case -** Follow the below given steps,
    0. **SampleMode** should be selected.
    1. **SelectCRCCCITT** as **Algorithm** from the list available.
    2. Under **Condition**, "**Error**" should be selected.
    3. Under **Input**, **Enter** the path of the file name to get its **CRC**. The file should be in "**.txt**" format which should not exceed 5000bytes.
    4. **Click** on **Run** button to execute. **Refresh** button can be used if new Inputs have to be given.

Sample Output

- **For No Error Case:** The **CalculatedCRC** should be **Zero** when the "**.txtfile**" is received by the **Node2**. The message "**DataFrame** is **Flowing** from **Node1** to **Node2** with **NoError**".
- **For Error Case:** The **CalculatedCRC** should be **Non-Zero** when the ".txt file" is received by the **Node2**. The message "**DataFrame** is **Flowing** from **Node1** to **Node2** with **Error**".

Error Detection Code - Longitudinal Redundancy Check (LRC)

Programming Guidelines

This section guides the user to link his/her own code for Longitudinal Redundancy Check to NetSim.

Pre - Conditions

The user program should read the inputted scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

The path of the input file and the output file can be viewed on clicking the Button "Path" in NetSim.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%" → NetSim → "Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and,

The results of the program should be written into the output file **Output.txt**.

Input File Format	Output File Format
'**Input.txt**' file has the original data bits, the type of parity (odd or even) and the error data bits. Parity=Odd (or Even) Data=Tetcos Data_Bits_Original=0101010001100101011101000110001 10110111101110011 Data_Bits_Error=0010101101000101010101000110001101 10111101110011	'**Output.txt** ' file, the First line is the LRC bits value for the original data bits. The Second line is the LRC bits value for the error data bits. 00111010 - LRC bits of the original data 01000101 - LRC bits of the error data

Interface Source Code

Interface source code written in C is given using this the user can write only the Longitudinal Redundancy Check inside the function fnLRC () using the variables already declared.To view the interface source code, go to

NetSim Installation path / src / Programming/ LRC.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7

- Open file location in Windows Vista
- Properties --> find target in Windows XP.

Sample Scenarios:

Objective - To study the working of Longitudinal Redundancy Check (LRC)

How to Proceed? - The objective can be executed in NetSim using the programming exercise available, under programming user has to select Error Detecting Codes in that Longitudinal redundancy Check,

Sample Input - In the Input panel the following steps need to be done,

- **SampleMode** should be **selected**.
- Parity need to be selected. Either Odd or Even can be selected.
- Data need to be entered. Maximum of 8 alphabets can be entered.
- Error Position needs to be selected. The value ranges from **1** to **48**.
- Then **Run** button need to be **clicked**. **Refresh** button can be used if new Inputs have to be given.

Output - The following steps are under gone internally,

- Data Bits which if in the binary form is obtained. This Data Bits is obtained for Original Data entered as well as for the Error selected.
- Depending on the Parity selected, tables of LRC of Original Data Bits and LRC of Error Data Bits are obtained.
- In each of the above mentioned tables last row would contain the Parity.
- Error In Column(s) is obtained.

Framing Sequence – Bit Stuffing

Programming Guidelines

This section guides the user to link his/her own code for Bit Stuffing to NetSim.

Pre - Conditions

The user program should read the input scenario from text file named '**Input**' with extension txt which is in **Temporary Directory**The user program after executing the concept should write the required output to a file named '**Output'** with extension txt in **Temporary Directory.**

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%" → NetSim → "Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and

The results of the program should be written into the output file **Output.txt**.

Input File	**Output File**
The "Input.txt" file contains, Destination Address=Value Source Address=Value Data=Message CRC Polynomial=Value Error Status=0 or 1 Seed Value=Value **Example:** Destination Address=00011111 Source Address=00111111	The "Output.txt" file contains, Message=Value> H=Value>e=Value>l=Value>l=Value>o=Value> Binary Values= Value > CRC Polynomial= Value > CheckSumSender= Value > <Stuffing> Destination Address= Value > Source Address= Value > Data= Value > <DeStuffing> Destination Address= Value > Source Address= Value >

Data=Hello CRC Polynomial=10010 Error Status=0 Seed Value=45	Data= Value > Error Status= Value > CheckSumReceiver= Value > Binary Values= Value > H= Value >e= Value >l= Value >l= Value >o= Value > Message= Value > **Example:** Message=Hello> H=72>e=101>l=108>l=108>o=111> Binary Values=0100100001100101011011000110110001101111> CRC Polynomial=10011> CheckSumSender=1110> <Stuffing> Destination Address=000111110> Source Address=001111101> Data=0100100001100101011011000110110001101111110110> <DeStuffing> Destination Address=00011111> Source Address=00111111> Data=0100100001100101011011000110110001101111111110> Error Status=0> CheckSumReceiver=0000> BinaryValues=0100100001100101011011000110110001101111> H=72>e=101>l=108>l=108>o=111> Message=Hello>

Interface Source Code

Interface source code written in C is given. Using this, the user can write only the functions **fnBitStuffing () and fnDeStuffing ()**, using the variables already declared.

To view the interface source code, go to **NetSim Installation path / src / Programming/ BitStuffing.c**

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista
- Properties --> find target in Windows XP.

Sample Scenarios:

Objective - To study the working of Bit Stuffing technique.

How to Proceed? - The objective can be executed in NetSim using the programming exercise available, under programming user has to select Framing Sequence → Bit Stuffing.

Sample Inputs - In the Input panel,

- Sample mode should be selected.
- Fill in the HDLC Frame fields available
  - Enter the 8 binary digits in the **Source Address** field.
  - Enter the 8 binary digits in the **Destination Address** field.
  - Enter data, with a maximum of 5 characters
  - CRC polynomial will be chosen by default and select either **Error** or **No Error.**
- Then **Run** button needs to be clicked. **Refresh** button can be used if new Inputs have to be given.

Output - The following steps are done internally by NetSim -

- Data will be converted into ASCII Values. ASCII Values will be converted into Binary Values.
- The Binary Value for CRC polynomial will be shown.
- Checksum will be calculated for the user data in **Sender** side.
- HDLC frame will be formed in Sender side and Bit Stuffing process is animated (Adding '0' for every consecutive **five** 1's).
- Then Destuffing process will be animated in **Receiver** side.
- Checksum will be calculated in receiver side.
- Again Binary Values will be converted into ASCII values.
- Finally the ASCII values will be converted into Data which the user entered.

Framing Sequence – Character Stuffing

Programming Guidelines

This section guides the user to link his/her own code for Character Stuffing to NetSim.

Pre - Conditions

The user program should read the input scenario from text file named '**Input**' with extension txt which is in **Temporary Directory**

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt in **Temporary Directory.**

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and

The results of the program should be written into the output file **Output.txt**.

Input File	**Output File**
The "Input.txt" file contains, Starting Delimiter=Value Destination Address=Value Source Address=Value Data=Value Checksum=Value Ending Delimiter=Value **Example:** Starting Delimiter=a Destination Address=eraerwbr	The "Output.txt" file contains, Stuffing> Destination Address=Value> Source Address= Value> Data= Value> Checksum= Value> DeStuffing> Destination Address= Value> Source Address= Value> Data= Value> Checksum= Value>

Source Address=asdasdas Data=sdfgf Checksum=shfsdfsd Ending Delimiter=h	**Example:** Stuffing> Destination Address=eraaerwbr> Source Address=aasdaasdaas> Data=sdfgf> Checksum=shhfsdfsd> DeStuffing> Destination Address=eraerwbr> Source Address=asdasdas> Data=sdfgf> Checksum=shfsdfsd>

Interface Source Code

Interface source code written in C is given. Using this, the user can write only the functions **fnCharacterStuffing () and fnDeStuffing ()**, using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ CharacterStuffing.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista
- Properties --> find target in Windows XP.

Sample Scenarios:

Objective - To study the working of Character Stuffing technique.

How to Proceed? - The objective can be executed in NetSim using the programming exercise available, under programming user has to select Framing Sequence → Character Stuffing.

Sample Input - In the Input panel the following steps need to be done,

- **SampleMode** should be **selected**.
- Fill in the HDLC Frame fields available.

- Starting Delimiter has to be filled in.
- Enter the 8 characters in the Destination Address field.
- Enter the 8 characters in the Source Address field.
- Enter in Data field with a maximum of 8 characters.
- Enter the 8 characters in the Check Sum field.
- Ending Delimiter has to be filled in.

- Then **Run** button need to be **clicked**. **Refresh** button can be used if new Inputs have to be given.

Output - The following steps are under gone internally,

- HDLC Frame will be formed in **Sender** side.
- Character stuffing process will be animated in Sender Side.
- Then destuffing process will be animated in Receiver side.
- Once the sample experiment is done **Refresh** button can be **clicked** to create new samples.

Virtual Scheduling Algorithm

Programming Guidelines

This section guides the user to link his/her own code for Virtual Scheduling Algorithm to NetSim.

Pre - Conditions

The user program should read the inputted scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

The path of the input file and the output file can be viewed on clicking the Button "Path" in NetSim.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%" → NetSim → "Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and,

The results of the program should be written into the output file **Output.txt**.

Input File Format	Output File Format
Input File Content PCR=1 CDVT=0.200 Actual_Arriva_Time=0.100 Previous_Theoratical_Time=0.000	This gives us the format of '**Output. Txt'** to which the user has to write his program Each value should be separated by a delimiter**'>'**. There should be no extra white spaces or blank lines. **Example:** Value1>Value2>Value3> Value1 – Cell conformation Flag, Value 2 – Next Cells Expected arrival Time Value 3 – Current cell's Actual arrival time. **Note:** The above convention to write into the **'Output.Txt'** are **mandatory**. A sample file format is as follows:3>2.000>1.000>

Interface Source Code

Interface source code written in C is given using this the user can write only the Longitudinal Redundancy Check inside the **void main ()** using the variables already declared.To view the interface source code, go to

NetSim Installation path / src / Programming/ VSA.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista
- Properties --> find target in Windows XP.

Sample Scenarios:

Objective - To study the working of Virtual Scheduling Algorithm.

How to Proceed? - The objective can be executed in NetSim using the programming exercise available, under programming user has to select Virtual Scheduling Algorithm,

Sample Input - In the Input panel the following steps need to be done,

- **SampleMode** should be **selected**.
- Select the Peak Cell Rate (PCR). Its value ranges from **1** to **10.**
- Select the Cell Delay Variation Tolerance (CDVT). Its value ranges from **0.1** to **0.9.**
- Enter in the Actual Arrival Time.
- Then **Run** button need to be **clicked**. **Refresh** button can be used if new Inputs have to be given.

Output - The following steps are under gone internally,

- A graph with the following information is obtained,
    - Cell Confirmation,
    - Actual Arrival Time,
    - Expected Time (TAT)
    - Whether the Cell has been discarded or successfully received. This discarding or receiving of the cell depends on the Actual Arrival Time entered. Go through the 3 rules given in this program.
- Once the sample experiment is done, then **Refresh** button can be **clicked** to create New Samples.

Address Mask

Programming Guidelines

This section guides the user to link his/her own code for Address mask to NetSim.

Pre - Conditions

The user program should read the input from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%" → NetSim → "Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and,

The results of the program should be written into the output file **Output.txt**.

File Format

Input File	Output File
This gives us the contents of the '**Input. Txt'** from which the user has to get the values for his program The Input File format IP Address> **Sample Input text Format** Let us consider how a given input (Data file and error index) is stored in the text and read. IP_Address=192.169.0.150 Prefix_Value=1	The Output File format Binary value of IP Address> Prefix value>Suffix value> Binary value of Address mask> Decimal value of Address mask>. **Sample Output text Format** 11000000 10101001 00000000 10010110 > 1>31> 10000000 00000000 00000000 0000000>128>0>0>0>

Interface Source Code

Interface source code written in C is given using this the user can write only thefnAddressMAsk () function using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ AddressMask.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista
- Properties --> find target in Windows XP.

Sample Scenarios:

Objective - To understand the concept of finding Address mask through programming.

Sample Inputs - In the Input panel the following steps need to be done,

- **SampleMode** should be **selected**.
- Select the IP Address
- Select the Prefix value
- Then **Run** button need to be **clicked**. **Refresh** button can be used if new Inputs have to be given.

Output - The following steps are done internally,

- Binary value of IP Address is obtained.
- Prefix, Suffix values are obtained
- Binary value of Address Mask is obtained
- Decimal value of Address Mask is obtained
- Once the sample experiment is done, then **Refresh** button can be **clicked** to create New Samples.

Binary Conversion

Programming Guidelines

This section guides the user to link his/her own code for Binary conversion to NetSim.

Pre - Conditions

The user program should read the input from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and,

The results of the program should be written into the output file **Output.txt**.

File Format

Input File Format	Output File Format
This gives us the contents of the '**Input. Txt'** from which the user has to get the values for his program The Input File format IP Address> **Sample Input text Format** IP_Address=192.168.0.100	The Output File format First Byte value>128>Quotient of First Byte/128>Remainder of First Byte/128> Previous remainder>64>Quotient of Previous remainder/64>Remainder of Previous remainder /64> Previous remainder>32>Quotient of Previous remainder/32>Remainder of Previous remainder /32> Previous remainder>16>Quotient of Previous remainder/16>Remainder of

	Previous remainder /16> Previous remainder>8>Quotient of Previous remainder/8>Remainder of Previous remainder /8> Previous remainder>4>Quotient of Previous remainder/4>Remainder of Previous remainder /4> Previous remainder>2>Quotient of Previous remainder/2>Remainder of Previous remainder /2> Previous remainder>1>Quotient of Previous remainder/1>Remainder of Previous remainder /1>. .same procedure for all bytes Binary value> **Sample Output text Format** 192>128>1>64> 64>64>1>0> 0>32>0>0> 0>16>0>0> 0>8>0>0> 0>4>0>0> 0>2>0>0> 0>1>0>0> 168>128>1>40> 40>64>0>40> 40>32>1>8> 8>16>0>8> 8>8>1>0> 0>4>0>0>

	0>2>0>0> 0>1>0>0> 0>128>0>0> 0>64>0>0> 0>32>0>0> 0>16>0>0> 0>8>0>0> 0>4>0>0> 0>2>0>0> 0>1>0>0> 100>128>0>100> 100>64>1>36> 36>32>1>4> 4>16>0>4> 4>8>0>4> 4>4>1>0> 0>2>0>0> 0>1>0>0> 11000000 10101000 00000000 01100100 >

Interface Source Code

Interface source code written in C is given using this the user can write only the fnBinaryConversion () function using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ BinaryConversion.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista

- Properties --> find target in Windows XP.

Sample Scenarios:

Objective - To understand the concept of Binary conversion through programming.

Sample Inputs - In the Input panel the following steps need to be done,

- **SampleMode** should be **selected**.
- Select the IP Address
- Then **Run** button need to be **clicked**. **Refresh** button can be used if new Inputs have to be given.

Output - The following steps are done internally,

- Each bit is obtained one by one.
- Binary value of IP Address is obtained.
- Once the sample experiment is done, then **Refresh** button can be **clicked** to create New Samples.

Classless InterDomain Routing

Programming Guidelines

This section guides the user to link his/her own code for Classless Inter Domain Routing to NetSim.

Pre - Conditions

The user program should read the inputted scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

The path of the input file and the output file can be viewed on clicking the Button "Path" in NetSim.

Note:

The temporary directory is navigated through the following step.

Run →Type **"%temp%" → NetSim → "Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and,

The results of the program should be written into the output file **Output.txt**.

File Format

Input File Format	**Output File Format**
This gives the contents of the '**Input. Txt'** from which the user has to get the values. **The Input File contains:** Starting_IP_Address=150.0.0.0 No_of_Networks=3 No_of_Hosts_in_Network_1=512 No_of_Hosts_in_Network_2=1024 No_of_Hosts_in_Network_3=2048 Host_IP_Address=150.0.1.2	The Output file contains the First address, Last address, Mask, Value and Network Address of all the networks. First Address: The Starting Address of the Network Last Address: The last address in the Network. Mask: The Mask address (Mask should be 4 byte format). Number of Node: The number of nodes in the network. Value: The value will be either 0 or 1. 0 - The next line has the network address to which the given host belongs. 1 - Means that the host does not belong to any network.

	Network Address: If the Value is 1 then the network to which the host belongs is given. **Note:** Each data will have a separate line and each line will end with ">" **A Sample Output File Format** 150.0.0.0> 150.0.1.255> 255.255.254.0> 512> 150.0.4.0> 150.0.7.255> 255.255.252.0> 1024> 150.0.8.0> 150.0.15.255> 255.255.248.0> 2048> 0> 150.0.0.0> For each network there will be first address, last address, mask and number of hosts. Since the input is 3 networks, there are 3 set of outputs. The last two lines give the details about whether the host is present in any of the network.

Interface Source Code

Interface source code written in C is given using this the user can write only the CIDR inside the function fnCIDR () using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ CIDR.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista
- Properties --> find target in Windows XP.

Sample Scenarios:

Objective - To Implement Classless InterDomain Routing (CIDR).

How to Proceed? - The objective can be executed in NetSim using the programming exercise available, under programming user has to select Classless InterDomain Routing,

Sample Input - By using the **Input Panel** that is available on the left hand side a **Sample Scenario** is created. The Steps involved in creating are as follows,

- **SampleMode** should be **selected**.
- **Select** the **Starting IP Address** from the given range.
- **Select** the **No. of Networks** from the given range. Maximum of **6** networks can be selected.
- **Select** the **No. of Hosts** from the given range.
- **Click** on **Add** button to add the **No.ofHosts** onto the Hosts Field. The use of **Add** button depends on the **No. of Networks** selected. If a new No. of Hosts has to be added then remove button can be used.
- Select the **Host IP Address**.
- Then **Run** button need to be **clicked**. **Refresh** button can be used if new Inputs have to be given.

Output - The following steps are under gone internally,

- Based on the Starting IP Address a table is obtained.
- There are 4 columns available in the output table, i.e. First Address, Last Address, CIDR Mask and No. of Hosts.

- The First Address of the first network would be nothing but the Starting IP Address which is selected. No. of rows in the table depends on the No. of Networks selected.
- The Last Address depends on depends on the No. of Host selected.
- CIDR Mask is obtained internally by using the following formula,

  CIDR Mask = 32 - (log (No. of Host) / log (2))

- Once the sample experiment is done, then **Refresh** button can be **clicked** to create New Samples.

Network Address

Programming Guidelines

This section guides the user to link his/her own code for Network Addresses to NetSim.

Pre - Conditions

The user program should read the input from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and,

The results of the program should be written into the output file **Output.txt**.

File Format

Input File	Output File

IP_Address=192.168.0.140 Prefix_Value=1	The Output File format Binary value of IP Address> Prefix value> Binary value of Address mask> Binary value of Network Address> Decimal value of Network Address>. **Sample Output text Format** 11000000 10101000 00000000 10001100 >1> 10000000 00000000 00000000 00000000> 10000000 00000000 00000000 00000000> 128>0>0>0>

Interface Source Code

Interface source code written in C is given using this the user can write only the fnNetworkAddress () function using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ NetworkAddress.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista
- Properties --> find target in Windows XP.

Sample Scenarios:

Objective - To understand the concept of finding Network Address through programming.

Sample Inputs - In the Input panel the following steps need to be done,

- **SampleMode** should be **selected**.
- Select the IP Address

- Select the Prefix value
- Then **Run** button need to be **clicked**. **Refresh** button can be used if new Inputs have to be given.

Output - The following steps are done internally,

- Binary value of IP Address is obtained.
- Binary value of Address Mask is obtained
- Binary value of Network Address is obtained
- Decimal value of Network Address is obtained
- Once the sample experiment is done, then **Refresh** button can be **clicked** to create New Samples.

Special Addresses

Programming Guidelines

This section guides the user to link his/her own code for Special Addresses to NetSim.

Pre - Conditions

The user program should read the input scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%" → NetSim → "Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and,

The results of the program should be written into the output file **Output.txt**.

File Format

Input File	Output File
IP_Address=192.168.0.160 Prefix_Value=5	The Output File format Binary value of IP Address> 32>Prefix value>Suffix value> Binary value of Prefix Part> Binary value of Suffix Part> Condition number in the table>condition> Type of address>. **Sample Output text Format** 11000000 10101000 00000000 10100000 > 32>5>27> 11000>000 10101000 00000000 10100000> It doesn't Satisfy the four conditions> Not a special address>

Interface Source Code

Interface source code written in C is given using this the user can write only the fnSpecialAddress () function using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ HammingCode.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista
- Properties --> find target in Windows XP.

Sample Scenarios:

Objective - To understand the concept of Special Addresses through programming.

Sample Inputs - In the Input panel the following steps need to be done,

- **SampleMode** should be **selected**.
- Select the IP Address in slash notation
- Then **Run** button need to be **clicked**. **Refresh** button can be used if new Inputs have to be given.

Output - The following steps are done internally,

- Binary value of IP Address is obtained.
- Binary value of Address Mask is obtained
- Prefix, Suffix values are obtained
- Binary value of Prefix part and Suffix part are obtained
- Type of address is obtained
- Once the sample experiment is done, then **Refresh** button can be **clicked** to create New Samples

Subnetting

Programming Guidelines

This section guides the user to link his/her own code for Sub-netting to NetSim.

Pre - Conditions

The user program should read the inputted scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

The path of the input file and the output file can be viewed on clicking the Button "Path" in NetSim.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and,

The results of the program should be written into the output file **Output.txt**.

File Format

Input File Format	Output File Format
Class=B Network_Address=128.0.0.0 No_of_Subnets=2 Host_IP_Address=128.0.0.89	The Output file contains the Default mask (in decimal), Network portion of Default mask (in binary), Host portion of Default mask (in binary), Subnet mask bit value, Network portion of Subnet mask (in binary), Host portion of Subnet mask (in binary), Subnet mask (in decimal), number of zero in host portion, number of host in each subnet, Host address, subnet number of the given host, subnet address of the given host and Subnet address, Starting address, Ending address, Broadcast address of each subnet. The Output File format Default mask (in decimal)> Network portion of Default mask (in binary)>Host portion of Default mask (in binary)> Subnet mask bit value> Subnet mask (in decimal)> Network portion of Subnet mask (in binary)>Host portion of Subnet mask (in binary)> Number of zero in host portion,>Number of host in each subnet> Host address>Subnet number of the given host >Subnet address of the given host> Subnet address>Starting address>Ending address>Broadcast

	address > . . .until number of subnet is reached Sample Output text Format 255.255.0.0 11111111 11111111>00000000 00000000> 1> 255.255.128.0> 11111111 11111111 1>0000000 00000000> 15>32766> 128.0.0.89>1>128.0.0.0> 1>128.0.0.0>128.0.0.1>128.0.127.254>128.0.127.255> 2>128.0.128.0>128.0.128.1>128.0.255.254>128.0.255.255>

Interface Source Code

Interface source code written in C is given using this the user can write only the Leaky bucket algorithm inside the function fnSubnet () using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ Subnet.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista
- Properties --> find target in Windows XP.

Sample Scenarios:

Objective - To understand the concept of Subnetting through programming.

How to Proceed? - The objective can be executed in NetSim using the programming exercise available, under programming user has to select Subnetting,

Sample Inputs - In the Input panel the following steps need to be done,

- **SampleMode** should be **selected**.
- Select the Class Name
- Select the Network Address from the given list.
- Select the Number of Subnets from the given list.
- Select the Host IP Address.
- Then **Run** button need to be **clicked**. **Refresh** button can be used if new Inputs have to be given.

Output - The following steps are under gone internally,

- Subnet mask value is obtained.
- Number of host in the each subnet is obtained
- Subnet address, Starting address, Ending address and Broadcast address are obtained.
- Subnet address of the given Host IP address is obtained.
- Once the sample experiment is done, then **Refresh** button can be **clicked** to create New Samples.

EUI-64 Interface Identifier

Programming Guidelines

This section guides the user to link his/her own code for EUI 64 Interface Identifier to NetSim.

Pre - Conditions

The user program should read the inputted scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and,

The results of the program should be written into the output file **Output.txt**.

File Format

Input File	Output File
IPV6_Network_Id=2000:FE21:5931:72 C3 MAC_Address=11-11-FF-FF-12-34	The Output File format MAC Address> First part of MAC Address>Second part of MAC Address FF-EE appended Address> First byte value>Binary value of First byte> 7th bit value>Complement value of 7th bit Complemented binary value of First byte>Hexa decimal value of complemented binary value> Interface Id value> Interface Id in colon notation> IPV6 prefix value>

	IPV6 Address> Sample Output text Format 11-11-FF-FF-12-34> 11-11-FF>FF-12-34> 11-11-FF-FF-FE-FF-12-34> 11>00010001> 0>1> 00010011>13> 13-11-FF-FF-FE-FF-12-34>1311:FFFF:FEFF:1234> 2000:FE21:5931:72C3> 2000:FE21:5931:72C3:1311:FFFF:FEFF:1234>

Interface Source Code

Interface source code written in C is given using this the user can write only the fnEUI64 () function using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ IPV6EUI64.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista
- Properties --> find target in Windows XP.

Sample Scenarios:

Objective - To understand the concept of EUI 64 Interface Identifier through programming.

How to Proceed? - The objective can be executed in NetSim using the programming exercise available, under programming user has to select IPV6 Addressing → EUI 64 Interface Identifier,

Sample Inputs - In the Input panel the following steps need to be done,

- **SampleMode** should be **selected**.
- Select the IPV6 Network Id (IPV6 Prefix)
- Enter the MAC Address.

- Then **Run** button need to be **clicked**. **Refresh** button can be used if new Inputs have to be given.

Output - The following steps are under gone internally,

- MAC Address is divided into two parts
- FF-FE value is appended between the two parts
- Interface Identifier is found by complementing 7^{th} bit of firs byte
- IPV6 Network Id (IPV6 prefix) and Interface Identifier are combined to produce IPV6 address
- Once the sample experiment is done, then **Refresh** button can be **clicked** to create New Samples.

IPV6 Host Addresses

Programming Guidelines

This section guides the user to link his/her own code for IPV6 Host Addresses to NetSim.

Pre - Conditions

The user program should read the inputted scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and,

The results of the program should be written into the output file **Output.txt**.

File Format

Input File Format	Output File Format
IPV6_Address=1111:2222: 3333:4444:AAAA:BBBB: CCCC:DDDD Prefix_Length=12	The Output File format IPV6 Address> Binary value of IPV6 Address> Prefix part of IPV6 Address>Suffix part of IPV6 Address> Prefix part of Starting Address>Suffix part of Starting Address> Starting Address in hexa decimal notation> Prefix part of Ending Address>Suffix part of Ending Address> Ending Address in hexa decimal notation> Sample Output text Format 1111:2222:3333:4444:AAAA:BBBB:CCCC:DDDD> 00010001000100010010001000100010011001100110011 01000100010001001010101010101010101011101110111011 11001100110011001101110111011101>12>116> 000100010001>00010010001000100010001100110011001 10100010001000100101010101010101010111011101110 11100110011001100110111011101101> 000100010001>00000000000000000000000000000000000 00000000000000000000000000000000000000000000000000 00000000000000000000000000000000> 1110:0000:0000:0000:0000:0000:0000:0000> 000100010001>11111111111111111111111111111111111 11111111111111111111111111111111111111111111111111 11111111111111111111111111111111> 111F:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF>

Interface Source Code

Interface source code written in C is given using this the user can write only the fnHostAddresses () function using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ IPV6HostAddress.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista
- Properties --> find target in Windows XP.

Sample Scenarios:

Objective - To understand the concept of IPV6 Host Addresses through programming.

How to Proceed? - The objective can be executed in NetSim using the programming exercise available, under programming user has to select IPV6 Addressing →IPV6 Host Addresses,

Sample Inputs - In the Input panel the following steps need to be done,

- **SampleMode** should be **selected**.
- Enter the IPV6 Address.
- Select the Prefix length.
- Then **Run** button need to be **clicked**. **Refresh** button can be used if new Inputs have to be given.

Output - The following steps are under gone internally,

- IPV6 is separated into two parts: Network Id bits (prefix) and Host Id bits (suffix)
- Starting address of the network is found by replacing each bits of suffix part with zero
- Ending address of the network is found by replacing each bits of suffix part with one
- Once the sample experiment is done, then **Refresh** button can be **clicked** to create New Samples.

IPV6 Subnetting

Programming Guidelines

This section guides the user to link his/her own code for IPV6 Subnetting to NetSim.

Pre - Conditions

The user program should read the inputted scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%" → NetSim → "Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and,

The results of the program should be written into the output file **Output.txt**.

File Format

Input File Format	Output File Format
IPV6_Address=1111:2222:3333:4444:AAAA:BBBB:CCCC:DDDD Prefix_Length=12 Number_of_Subnets=2 Number_of_Level=2 Subnets_in_Level1=2 Subnets_in_Level2=4	The Output File format Binary value of IPV6 Address> Prefix part of IPV6 Address>Suffix part of IPV6 Address> Number of subnets>number of subnet msk bit> Prefix part of IPV6 Address>Subnet Id part of IPV6 address>Suffix part of IPV6 Address> Prefix part of IPV6 Address>Subnet Id part of IPV6 address>Suffix part of IPV6 Address> Prefix part of Level 1 's first subnet Address> Subnet Id part of Level 1 's first subnet Address> Suffix part of Level 1 's first subnet Address> Hexa decimal value of Level 1 's first

	subnet address>prefix length of Level 1 ‘s first subnet> . . until number of subnet reached in the first level First level’s subnet number>Prefix part of Level 2 ‘s first subnet Address> Subnet Id part of Level 2 ‘s first subnet Address> Suffix part of Level 2 ‘s first subnet Address> Hexa decimal value of Level 2 ‘s first subnet address>prefix length of Level 2 ‘s first subnet> . . until the number of subnets reached in the second level of first level’s subnet . . until the number of subnets reached in the first level Sample Output text Format 1111:2222:3333:4444:AAAA:BBBB:CCCC:DDDD> 000100010001100100100010001000100011001100110100010001000100100101010101010101010101110111011011101111001100110011001101110111011101> 000100010001> 000100010001>0> 000100010001>0>1111:2222:3333:4444:AAAA:BBBB:CCCC:DDDD>13> 000100010001>1> 000100010001>1>1119:2222:3333:4444:AAAA:BBBB:CCCC:DDDD>13> 0>1111:2222:3333:4444:AAAA:BBBB:CCCC:DDDD>15> 0>1113:2222:3333:4444:AAAA:BBBB:CCCC:DDDD>15> 0>1115:2222:3333:4444:AAAA:BBBB:CCCC:DDDD>15> 0>1117:2222:3333:4444:AAAA:BBBB:CCCC:DDDD>15> 1>1119:2222:3333:4444:AAAA:BBBB:CCCC:DDDD>15>

	1>111B:2222:3333:4444:AAAA:BBBB:CCCC:DDDD>15> 1>111D:2222:3333:4444:AAAA:BBBB:CCCC:DDDD>15> 1>111F:2222:3333:4444:AAAA:BBBB:CCCC:DDDD>15>

Interface Source Code

Interface source code written in C is given using this the user can write only the fnIPV6Subnetting () function using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ IPV6Subnetting.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista
- Properties --> find target in Windows XP.

Sample Scenarios:

Objective - To understand the concept of IPV6 Subnetting through programming.

How to Proceed? - The objective can be executed in NetSim using the programming exercise available, under programming user has to select IPV6 Addressing →IPV6 Subnetting,

Sample Inputs - In the Input panel the following steps need to be done,

- **SampleMode** should be **selected**.
- Enter the IPV6 Address.
- Select the Prefix length.
- Select the number of subnets value
- Click the Add value button to add the levels
- Then **Run** button need to be **clicked**. **Refresh** button can be used if new Inputs have to be given.

Output - The following steps are under gone internally,

- IPV6 is separated into two parts: prefix and suffix
- Number of subnet mask id is calculated
- Then Subnet Id portion is derived from suffix part
- Using the subnet id portion, subnets are calculated
- Like that, subnets in the all levels are calculates by using the above steps
- Once the sample experiment is done, then **Refresh** button can be **clicked** to create New Samples.

Leaky Bucket Algorithm

Programming Guidelines

This section guides the user to link his/her own code for Leaky Bucket Algorithm to NetSim.

Pre - Conditions

The user program should read the inputted scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

The path of the input file and the output file can be viewed on clicking the Button "Path" in NetSim.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%" → NetSim → "Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and,

The results of the program should be written into the output file **Output.txt**.

File Format

Input File	Output File
Output_Capacity=100 Buffer_Size=100 Input_Capacity=100,100,500,100,100,100,100,100,100,100,	'Output.txt' file contains three lines of data. The data format in the output file should be as follows: Output rate at 1st second >Output rate at 2nd second>……..> Discard rate at 1st second >Discard rate at 2nd second>……..> Total number of seconds taken > The data should be stored in the file with a delimiter ">" in between. Sample File data 100>100>100>100>100>100>100>100>100>100>100> 0>0>300>0>0>0>0>0>0>0>0> 11>

Interface Source Code

Interface source code written in C is given using this the user can write only the Leaky bucket algorithm inside the function fnLBA() using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ LBA.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista
- Properties --> find target in Windows XP.

Sample Scenarios:

Objective - To understand the concept of Leaky Bucket Algorithm(LBA) through programming.

How to Proceed? - The objective can be executed in NetSim using the programming exercise available, under programming menu user has to select Leaky Bucket Algorithm.

Sample Inputs - In the Input panel the following steps need to be done,

- **SampleMode** should be **selected**.
- Select the Output Capacity from the given list of values. The value ranges from **100** to **1000.**
- Select the Size of Buffer from the given list of values. The value ranges from **100** to **1000.**
- Enter in the Input Capacity in the fields provided. The range that can be entered is from **100** to **1000.** These values will be plotted in the graph on the right hand side panel.
- Then **Run** button need to be **clicked**. **Refresh** button can be used if new Inputs have to be given.

Output - The following steps are under gone internally,

- Once the **Run** button is clicked a graph with Output and Discard Rate is obtained.
- The graph explains in detail as in what is the Output and Discard Packet Rate for the given inputs.
- Once the sample experiment is done, then **Refresh** button can be **clicked** to create New Samples.

Multi Level Multi Access

Programming Guidelines

This section guides the user to link his/her own code for Multi-Level Multi Access Protocol to NetSim.

Pre - Condition

User written program should read the value from the '**Input.txt**' in the temporary directory which is having input from the GUI at runtime

The output should be stored in '**Output.txt**' in the temporary directory for display.

User written program should return an integer value.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and, The results of the program should be written into the output file **Output.txt**.

Note: The naming of the input and the output file must be same as the text displayed in the Methodology screen

File Format

Input File	Output File
Number_of_Nodes=2 Node1_Address=250 Node2_Address=500	'**Output.txt**' file has the decade's number, decade's type, and actual decades, node addresses **Decade No>Decade Type>Decade>aderess1>adress2>…** Sample Output: 0>0>0000100100> 1>1>0000000001> 2>2>0000000001>500>

	3>1>0000100000> 4>2>0000000001>250>

Interface Source Code

Interface source code written in C is given using this the user can write only the Multi-Level Multi Access Protocol inside the function fnMLMA () using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ MLMA.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista
- Properties --> find target in Windows XP.

Sample Scenarios:

Objective - To study the working of Multi Level Multi Access (MLMA).

How to Proceed? - The objective can be executed in NetSim using the programming exercise available, under programming user has to select Multi-Level Multi -Access Protocol.

Sample Inputs - In the Input panel the following steps need to be done,

- **SampleMode** should be **selected**.
- Number of Nodes need to be **selected**. The value ranges from **2** to **15**.
- Node Address has to be entered and added onto the Node Address field. The Number of Nodes selected and the Number of Addresses added into the Address field should match.
- Then **Run** button need to be **clicked**. **Refresh** button can be used if new Inputs have to be given.

Output - The following steps are under gone internally,

- An Output table is formed.

- On the right hand panel an animation would play. This shows that higher the address given to the node higher the priority that node gets. Hence the node with the higher address would transmit the data first.
- Once the sample experiment is done, then **Refresh** button can be **clicked** to create New Samples.

Code Division Multiple Access

Programming Guidelines

This section guides the user to link his/her own code for Code Division Multiple Access to NetSim.

Pre - Conditions

The user program should read the input scenario from text file named '**Input**' with extension txt which is in **Temporary Directory.**

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt in **Temporary Directory.**

Note: The Temporary Directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and

The results of the program should be written into the output file **Output.txt**.

File Format

Input File Format	Output File Format
The “Input.txt” file contains, **NumberofTransmittingMobilestations=Value>** **MobileStation=Value>Data=Value>Code=Value>** **MobileStation=Value>Data=Value>Code=Value>** **MobileStation=*Value*> Data=Value>Code=Value>** **Example:** NumberOfTransmittingMobileStations=2> MobileStation=1>Data=10101010> Code=1 1 1 1 1 1 1 1> MobileStation=2>Data=11110000> Code=1 -1 1 -1 1 -1 1 -1>	**No. of Transmitting MobileStations=Value>** **MobileStation=Value>Data=Value>code=Value> Encode=Value>Signal=Value>** **MobileStation=Value>Data=Value>code=Value> Encode=Value>Signal=Value>** **MobileStation=Value>Data=Value>code=Value> Enecode=Value>Signal=Value** **Interference Pattern=Value>** **MobileStation=Value>code=Value>Decode=Value>Data=Value>** **MobileStation=Value>code=Value>Decode=Value>Data=Value>** **MobileStation=Value>code=Value>Decode=Value>Data=Value>** **Example:** NumberofMobileStations=2> MobileStation=1>Data=1 0 1 0 1 0 1 0 >Code=1 1 1 1 1 1 1 1 >Encode=1 -1 1 -1 1 -1 1 -1 >Signal=1 1 1 1 1 1 1 1 -1 -1 -1 -1 -1 -1 -1 -1 1 1 1 1 1 1 1 1 -1 -1 -1 -1 -1 -1 -1 -1 1 1 1 1 1 1 1 1 -1 -1 -1 -1 -1 -1 -1 -1 1 1 1 1 1 1 1 1 -1 -1 -1 -1 -1 -1 -1 -1 > MobileStation=2>Data=1 1 1 1 0 0 0 0 >Code=1 -1 1 -1 1 -1 1 -1 >Encode=1 1 1 1 -1 -1 -1 -1 >Signal=1 -1 1 -1 1 -1 1 -1

Interface Source Code

Interface source code written in C is given.Using this, the user can write only the function **fnEncode ()** and **fnDecode ()**, using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ CDMA.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista
- Properties --> find target in Windows XP.

Sample Scenarios:

Objective - To study the working of Code Division Multiple Access technique

How to Proceed? - The objective can be executed in NetSim using the programming exercise available. Under programming select Multiple Access Technology → Code Division Multiple Access.

Sample Inputs - In the Input panel the following steps need to be done,

- **Sample Mode** should be selected
- In default, the **Encode** operation will be enabled.
- Choose the **Number of Transmitting Mobile Stations.**
- Enter the **Binary Datas** as per the limits.
- Choose the respective **Codes** for each Mobile Station.

Number of Transmitting Mobile Stations →3 (Maximum of 8 Mobile Stations)

Mobile Station	Binary Data(1Byte)	Code
1	01110111	1 1 1 1 1 1 1 1
2	00110011	1 -1 1 -1 1 -1 1 -1

3	11011101	1 1 -1 -1 1 1 -1 -1

- Then **Run** button needs to be **clicked**. **Refresh** button can be used, if new inputs have to be given.

Output -The Output for the above sample is as follows,

- In **Transmitter side**, the values of each Mobile Station's **Binary Data, Code, Encode and Signal** are shown, if the user clicks the Mobile Station.
- The **Signals** will be broadcast from each Mobile Stations.
- The **Interference Pattern** will be formed in **Base Station** and if the user clicks the Base Station, interference Pattern will be displayed.
- **Decode mode** needs to be **selected.**
- In **Receiver side**, the **Interference Pattern** will be broadcasted to each **Mobile Station** from the **Base Station**.
- In **Receiver side,** the values of each Mobile Station's **Interference Pattern**, **Code, Decode and Binary Data** are shown, if the user clicks the Mobile Station.
- Once the sample experiment is done **Refresh** button can be **clicked** to create new samples.

The Values are showed in Table.

Transmitter Side

Mobile Station	Binary Data(1Byte)	Code	Encode	Signal
1	01110111	1 1 1 1 1 1 1 1	-1, 1, 1, 1, -1, 1, 1, 1	-1 -1 -1 -1 -1 -1 -1 -1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 -1 -1 -1 -1 -1 -1 -1 -1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
2	00110011	1 -1 1 -1 1 -1 1 -1	-1, -1, 1, 1, -1, -1, 1, 1	-1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1

				1 -1 1 -1 1 -1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1
3	11011101	1 1 -1 -1 1 1 -1 -1	1, 1, -1, 1, 1, 1, -1, 1	1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 -1 -1 1 1 -1 -1 1 1 1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 -1 -1 1 1 -1 -1 1 1 1 1 -1 -1

In Base Station

Mobile Station	Signal
1	-1 -1 -1 -1 -1 -1 -1 -1 1 1 1 1

	1 -1 3 1 3 1 1 -1 3 1 1 -1 -1 1 -3 -1 -1 1 -3 -1 1 3 -1 1 1 3 -1 1 1 -1 3 1 1 -3 1 3 1 1 -1 3 1 1 -1			
3	1 1 -3 -1 -1 1 -3 -1 1 3 -1 1 1 3 -1 1 1 -1 3 1 1 -1 3 1 3 1 1 -1 3 1 1 -1 -1 1 -3 -1 -1 1 -3 -1 1 3 -1 1 1 3 -1 1 1 -1 3 1 1 -3 1 3 1 1 -1 3 1 1 -1	1 1 -1 -1 1 1 -1 -1	8 8 -8 8 8 8 -8 8	11011101

Time Division Multiple Access

Programming Guidelines

This section guides the user to link his/her own code for Time Division Multiple Access to NetSim.

Pre - Conditions

The user program should read the input scenario from text file named '**Input**' with extension txt which is in **Temporary Directory.**

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt in **Temporary Directory.**

Note: The Temporary Directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and

The results of the program should be written into the output file **Output.txt**.

File Format

Input File Format	Output File Format
The “Input.txt” file contains, **Bandwidth=Value>No. of time slots=Value>Time slot Length=Value>Guard Interval=Value>** **Example:** BandWidth=100>TimeSlot=1>TimeSlotLength=100>GuardInterval=0.0>	**Guard Interval =Value>Bandwidth=Value>start time=Value >end time=Value>nouser=Value>** **channel no=Value>Bandwidth=Value>start time=Value>End time=Value>user=Value>** **Guard Interval =Value>Bandwidth=Value>start time=Value >end time=Value>nouser=Value>** **Guard Interval =Value>Bandwidth=Value>start time=Value >end time=Value>nouser=Value>** **channel no=Value>Bandwidth=Value>start time=Value>End time=Value>user=Value>** . . **and so on.** The value of the **GuardInterval** is **-1** and the value of the **no user** is **0** **Example:** GuardInterval=-1>Bandwidth=100>StartTime=0.0>EndTime=0.0>nouser=0> Channelno=1>Bandwidth=100>StartTime=0.0>EndTime=100.0>user=1> GuardInterval=-1>Bandwidth=100>StartTime=100.0>EndTime=100.0>nouser=0>

Interface Source Code

Interface source code written in C is given.Using this, the user can write only the function **fnTDMA()**, using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ TDMA.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista
- Properties --> find target in Windows XP.

Sample Scenarios:

Objective - To study the working of Time Division Multiple Access technique.

How to Proceed? - The objective can be executed in NetSim using the programming exercise available, under programming user has to select Multiple Access Technology → Time Division Multiple Access.

Sample Inputs - In the Input panel the following steps need to be done,

- **SampleMode** should be **selected**.
- Enter the Value of the **Bandwidth**.
- Enter the Value of the **No. of Time Slots.**
- Enter the Value of the **TimeSlotLength.**
- Enter the Value of the **GuardInterval**

Note:

The Values entered should be within the range.

Bandwidth(kHz)	200
No. of TimeSlots	3
TimeSlotLength(µs)	100
GuardInterval(µs)	10

- Then **Run** button need to be **clicked**. **Refresh** button can be used, if new Inputs have to be given.

Output -The Output for the above sample is as follows,

- Bandwidth is divided into Time slots called channel.
- The first channel is allocated to first mobile station, second channel is allocated to second mobile station and so on.
- The first Mobile Station will access the medium in first channel time slot.
- The Second Mobile Station will access the medium in second channel time slot, and so on.
- During the guard interval, No mobile station will access the medium. This is used to avoid the collision or interference.

Finally the table will be showed.

- Channel no. is obtained.
- Bandwidth value is obtained.
- Start Time and End Time of the Guard Interval is obtained
- Start Time and End Time of the channel is obtained.
- Which Mobile Station is accessing the allocated time slot is obtained.

Channel number	Bandwidth(kHz)	Time(µs)	Mobile Station
0	200	0.0 – 5.0	0
1	200	5.0 – 95.0	1
0	200	95.0 – 100.0	0
0	200	100.0 – 105.0	0
2	200	105.0 – 195.0	2
0	200	195.0 – 200.0	0
0	200	200.0 – 205.0	0

3	200	205.0 – 295.0	3
0	200	295.0 – 300.0	0
0	200	300.0 – 305.0	0
1	200	305.0 – 395.0	1
.			
.			
.			

Orthogonal Frequency Division Multiple Access

Programming Guidelines

This section guides the user to link his/her own code for Orthogonal Frequency Division Multiple Access to NetSim.

Pre-conditions

The user program should read the input scenario from text file named '**Input**' with extension txtwhich is in **Temporary Directory.**

The user program after executing the concept should write the required output to a file named '**Output'** with extension txtin **Temporary Directory.**

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and, the results of the program should be written into the output file **Output.txt**.

Input File Format	Output File Format
The following lines will be in Input.txt file. **Number of users =** *Value* **User1 =***Value* **User2 =***Value* **User3 =***Value* **User4 =***Value* **Number of sub carrier per user =** *Value* **Example:** Number of users = 4 User1 =111111111111 User2 =010101010101 User3 =101010101010 User4 =000000000000 Number of sub carrier per user = 4	**Parallel Data Conversion** **User 1** *Value* **User 2** *Value* **User 3** *Value* **User 4** *Value* **Bit Mapping** **User 1** *Value* **User 2** *Value* **User 3** *Value* **User 4** *Value* **SubCarrier Mapping** **User 1** *Value* **User 2** -*Value* **User 3** *Value* **User 4** *Value* **SubCarrier Addition** **User 1** *Value*

	User 2 *Value* **User 3** *Value* **User 4** *Value* **Example:** Parallel Data Conversion User 1 1 1 1 1 1 1 1 1 1 1 1 1 User 2 0 1 0 1 0 1 0 1 0 1 0 1 User 3 1 0 1 0 1 0 1 0 1 0 1 0 User 4 0 0 0 0 0 0 0 0 0 0 0 0 Bit Mapping User 1 1 1 1 1 1 1 1 1 1 1 1 1 User 2 -1 1 -1 1

	-1 1 -1 1 -1 1 -1 1 User 3 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 User 4 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 SubCarrier Mapping User 1 sin(2PIf0t) sin(2PIf1t) sin(2PIf2t) sin(2PIf3t) sin(2PIf0t) sin(2PIf1t) sin(2PIf2t) sin(2PIf3t) sin(2PIf0t) sin(2PIf1t) sin(2PIf2t) sin(2PIf3t) User 2 -sin(2PIf4t) sin(2PIf5t) -sin(2PIf6t) sin(2PIf7t) -sin(2PIf4t) sin(2PIf5t) -sin(2PIf6t) sin(2PIf7t) -sin(2PIf4t) sin(2PIf5t) -sin(2PIf6t) sin(2PIf7t) User 3 sin(2PIf8t) -sin(2PIf9t) sin(2PIf10t) -sin(2PIf11t) sin(2PIf8t) -sin(2PIf9t) sin(2PIf10t) -sin(2PIf11t) sin(2PIf8t) -sin(2PIf9t) sin(2PIf10t) -sin(2PIf11t) User 4 -sin(2PIf12t) -sin(2PIf13t) -sin(2PIf14t) -sin(2PIf15t) -sin(2PIf12t) -sin(2PIf13t) -sin(2PIf14t) -sin(2PIf15t) -sin(2PIf12t) -sin(2PIf13t) -sin(2PIf14t) -sin(2PIf15t) SubCarrier Addition User 1 sin 2PI(f0+f1+f2+f3)t sin 2PI(f0+f1+f2+f3)t sin 2PI(f0+f1+f2+f3)t

	User 2 sin 2PI(-f4+f5-f6+f7)t sin 2PI(-f4+f5-f6+f7)t sin 2PI(-f4+f5-f6+f7)t User 3 sin 2PI(f8-f9+f10-f11)t sin 2PI(f8-f9+f10-f11)t sin 2PI(f8-f9+f10-f11)t User 4 sin 2PI(-f12-f13-f14-f15)t sin 2PI(-f12-f13-f14-f15)t sin 2PI(-f12-f13-f14-f15)t

Interface Source Code

Interface Source code written in C is given. Using this, the user can writeonly the fnOFDMA() and fnSubcarrierAddition() functions, using the variables already declared.To view the interface source code, go to

NetSim Installation path/src/Programming/OFDMA.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista
- Properties --> find target in Windows XP.

Sample Scenarios:

Objective - To study the working of Orthogonal Frequency Division Multiple Access technique

How to Proceed? - The objective can be executed in NetSim using the programming exercise available. Under programming select Multiple Access Technology →OFDMA

Sample Inputs - In the Input panel,

- Sample Mode should be selected.
- The value of **Number of Users** needs to be selected.

- Enter the **Binary Data**for each user.
- **Number of SubCarriers per User** needs to be selected.
- Then **Run** button needs to be clicked. **Refresh** button can be used if new Inputs have to be given.

Output - The following steps are done internally by NetSim

- The values of each user**Binary Data, Parallel Conversion, Bit Mapping, SubCarrier Mapping**&**SubCarrier Addition and Transmission** will be shown in the right hand side of the screen by animation.
- Once the sample experiment is done **Refresh** button can be **clicked** to create new samples.

PC to PC Communication - Socket Programming TCP

Programming Guidelines

This section guides the user to link his/her own code for PC to PC Communication – Socket programming to NetSim.

Pre – Conditions

The user program should read the input scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

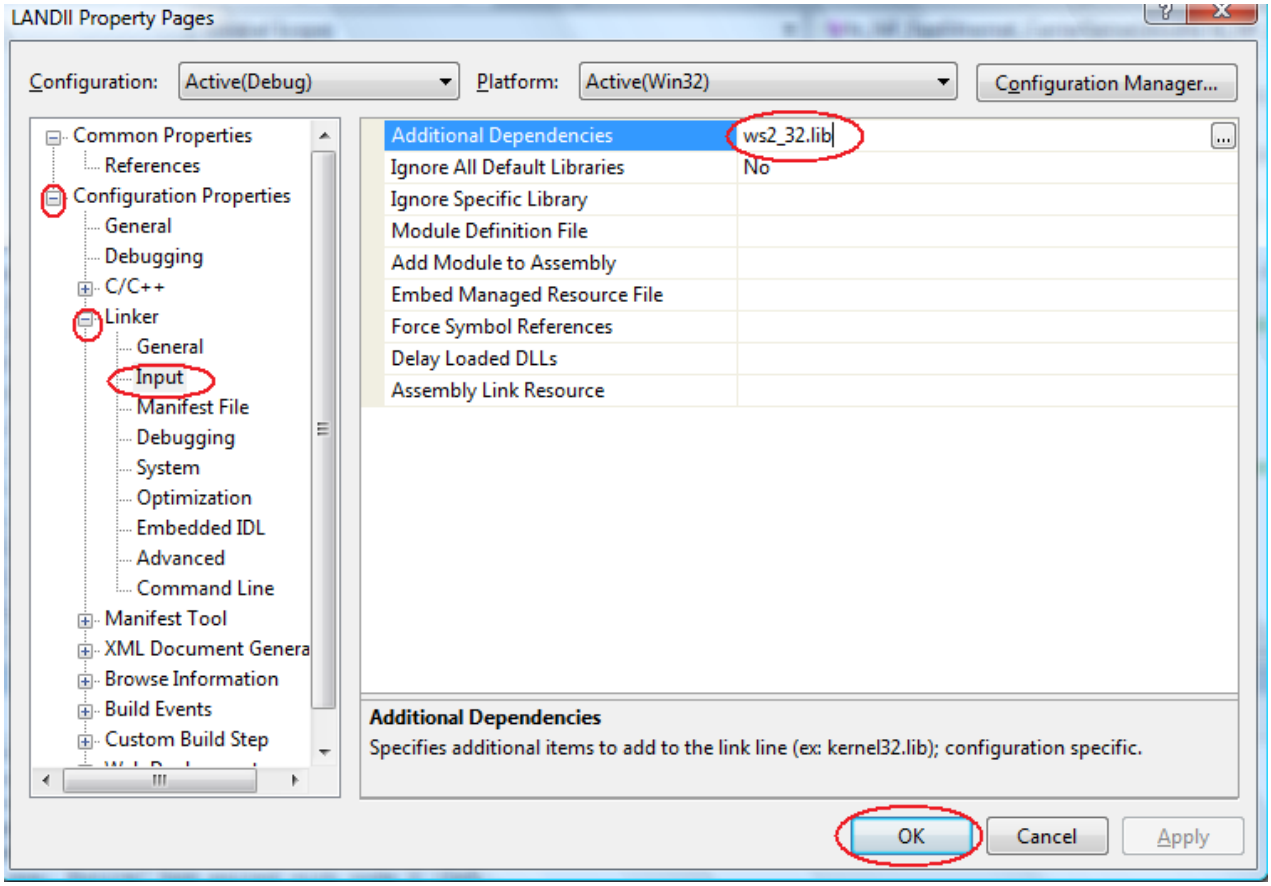
Executing the required concept and

The results of the program should be written into the output file **Output.txt**.

Coding Pre Requisites:

The prerequisite for the creation of sockets in the Windows-based C complier is they must have **winsock.h** header file in its standard header file folder. In order to use the commands of the winsock.h, there must be wsock32.lib library file present in the C complier. To link wsock32.lib, different steps have to be followed for different compilers as shown:

- For Visual Studio complier 2010 (10.0 versions), the library file linkage must be made in the following way:



a. Click **Project Menu** in the workspace.

b. Select **properties** in the pull-down menu or press Alt + F7

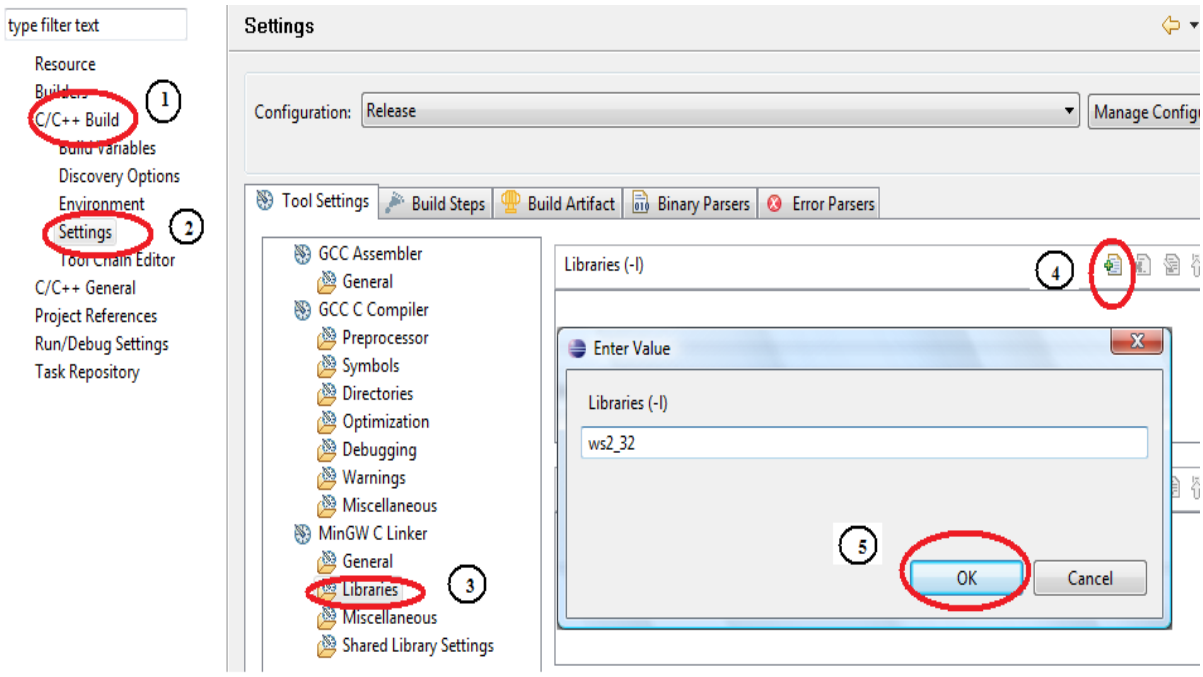
c. Then click **Linker.**

d. Select **Input**

d. Type **ws2_32.lib** file in the **Additional Dependencies** text box area.

e. Click **OK** to confirm the changes in the workspace.

- If you have an eclipse complier, the library linkage must be in the following way



a. Click **Project Menu** in the workspace.

b. Select **properties** in the pull-down menu

c. Then click C/C++ Build**.**

d. Select **Setting.**

e. Select **Libraries** under MinGW C Linker

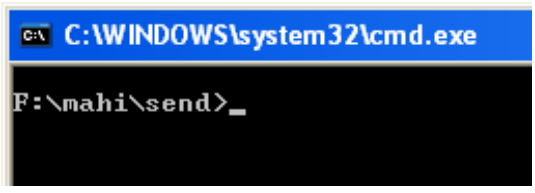
f. Press **add** button in the Libraries (-l) window

d. Type **ws2_32** file in the text box area.

e. Click **OK** to confirm the changes in the workspace.

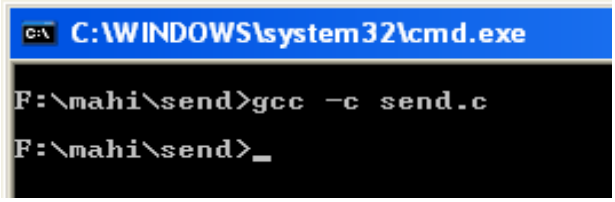
- If gcc compiler is used via command prompt:

a. Set the path to Socket program location.



b. Create output file (.o file) using below command.

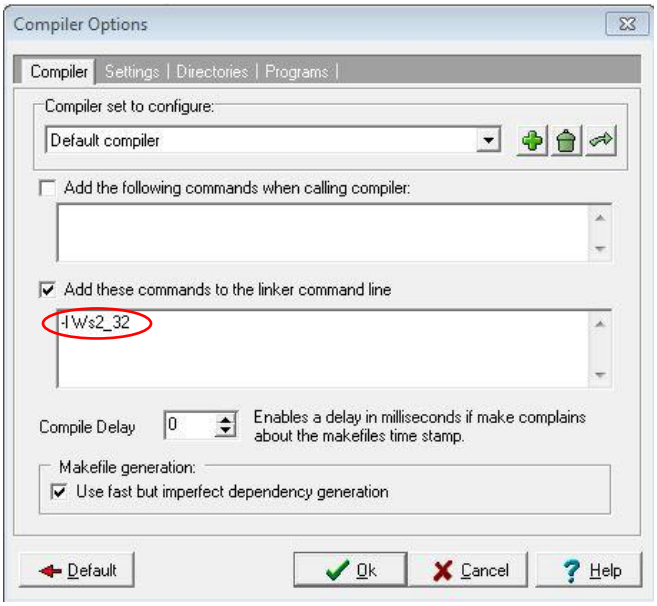
gcc -C Filename.c



c. Create exe file (.exe file) using below command

gcc -o send send.o -lws2_32

- If using Dev C++ IDE, go to Tools->compiler options and add -lws2_32 as follows before executing:



Input File Format	Output File Format
The data format in the input file is as follows, **Server Input** Protocol=TCP Operation=Server **Client** Protocol=TCP Operation=Client Destination IP=192.168.0.132	Write the received message Ex:192.168.0.2:- Requesting for the connection

<Client IP:-Message> Ex: 192.168.0.2:- Requesting for the connection	

Interface Source Code

Interface source code written in C is given using this the user can write only the TCP Client () function. To view the interface source code, go to

NetSim Installation path / src / Programming/ SocketTCP.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista
- Properties --> find target in Windows XP.

PC to PC Communication - Socket Programming UDP

Programming Guidelines

This section guides the user to link his/her own code for PC to PC Communication – Socket programming to NetSim.

Pre – Conditions`

The user program should read the input from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and

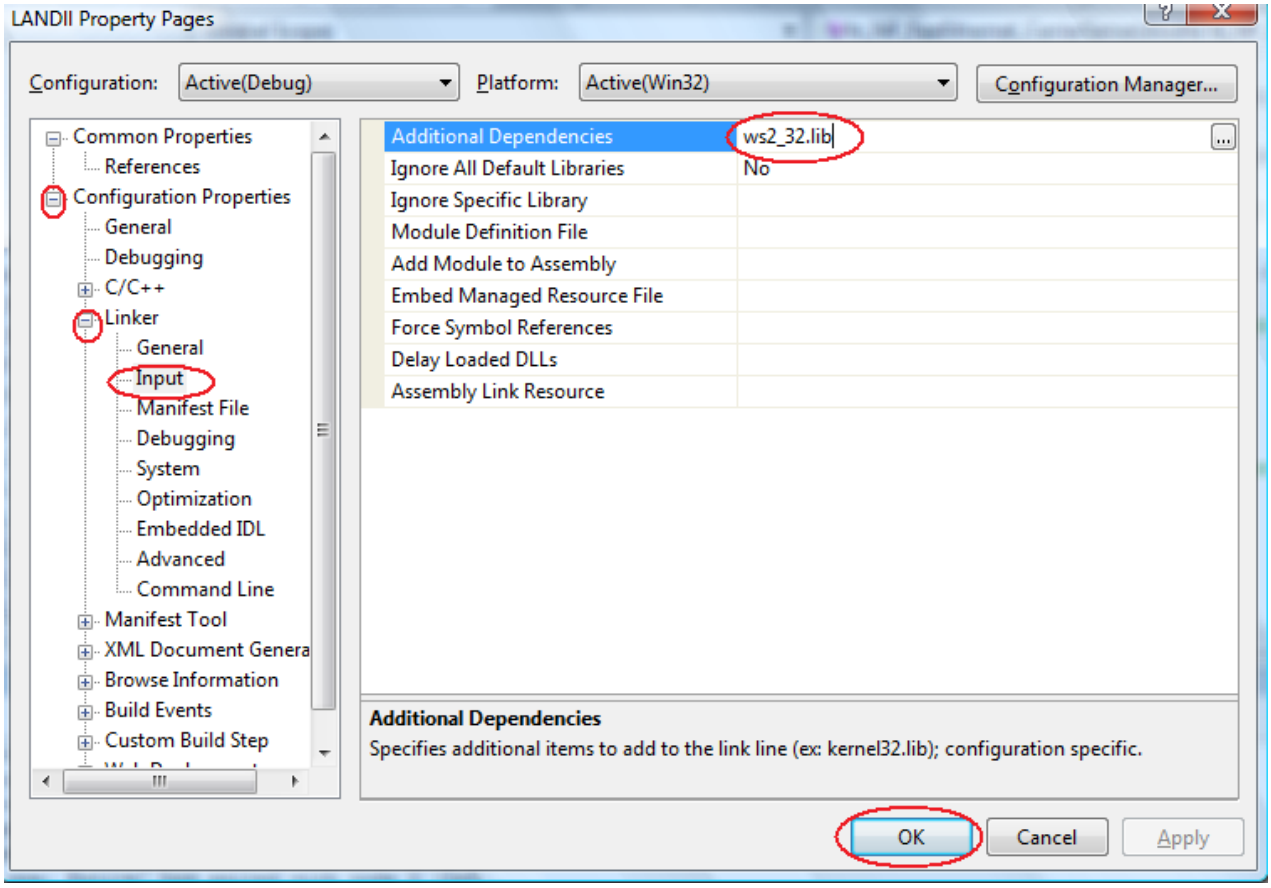
The results of the program should be written into the output file **Output.txt**.

Coding Pre Requisites:

The prerequisite for the creation of sockets in the Windows-based C complier is they must have **winsock.h** header file in its standard header file folder. In order to use the commands of the

winsock.h, there must be wsock32.lib library file present in the C complier. To link wsock32.lib, different steps have to be followed for different compilers as shown:

- For Visual Studio complier 2010 (10.0 versions), the library file linkage must be made in the following way:



a. Click **Project Menu** in the workspace.

b. Select **properties** in the pull-down menu or press Alt + F7

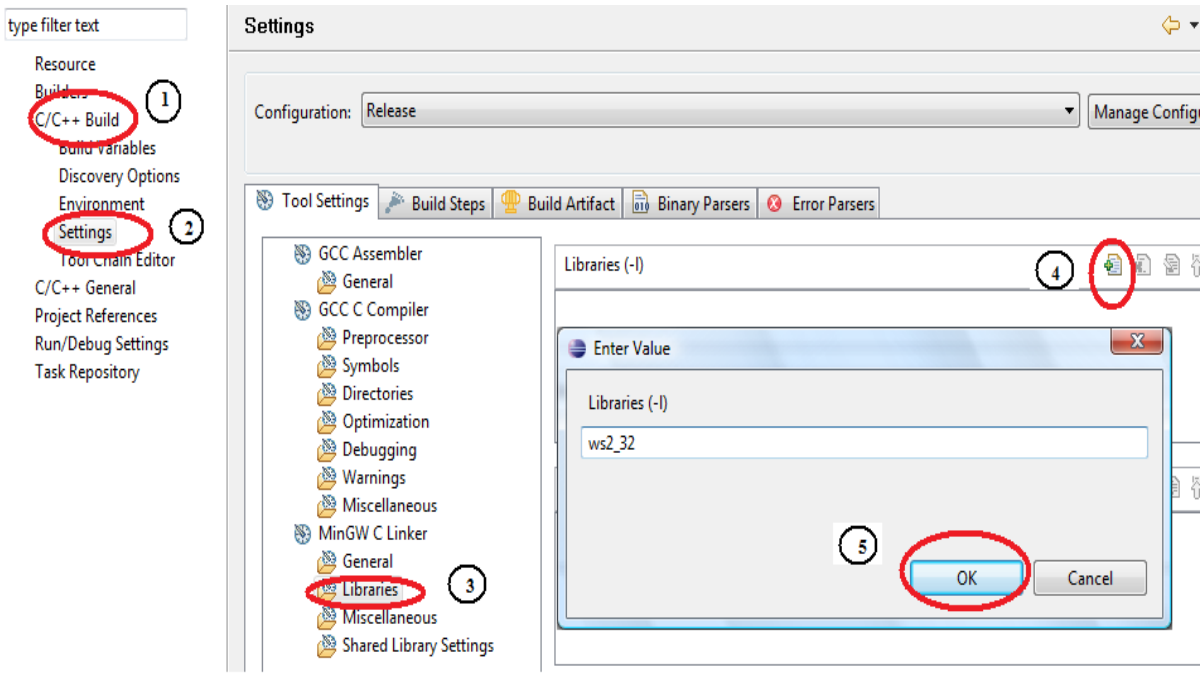
c. Then click **Linker.**

d. Select **Input**

d. Type **ws2_32.lib** file in the **Additional Dependencies** text box area.

e. Click **OK** to confirm the changes in the workspace.

- If you have an eclipse complier, the library linkage must be in the following way



a. Click **Project Menu** in the workspace.

b. Select **properties** in the pull-down menu

c. Then click C/C++ Build**.**

d. Select **Setting.**

e. Select **Libraries** under MinGW C Linker

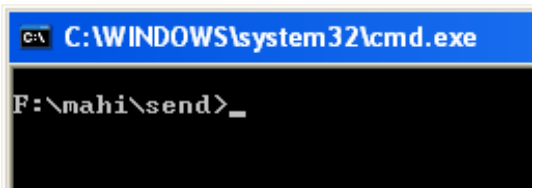
f. Press **add** button in the Libraries (-l) window

d. Type **ws2_32** file in the text box area.

e. Click **OK** to confirm the changes in the workspace.

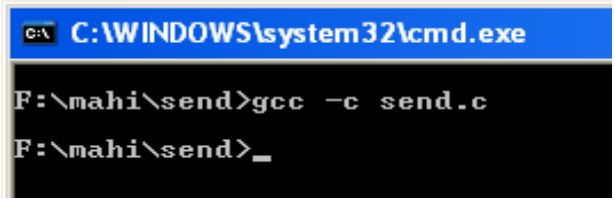
- If gcc compiler is used via command prompt:

a. Set the path to Socket program location.



b. Create output file (.o file) using below command.

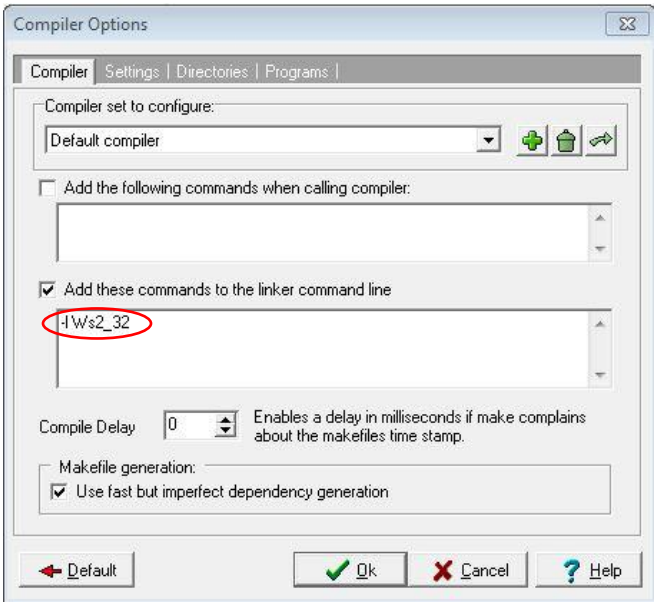
gcc -C Filename.c



c. Create exe file (.exe file) using below command

gcc -o send send.o -lws2_32

- If using Dev C++ IDE, go to Tools->compiler options and add -lws2_32 as follows before executing:



Input File Format	Output File Format
The data format in the input file is as follows,. **Server Input** Protocol= UDP Operation=Server **Client** Protocol=UDP Operation=Client Destination IP=192.168.0.132	Write the received message Ex:192.168.0.2:- Requesting for the connection

<Client IP:-Message> Ex: 192.168.0.2:- Requesting for the connection	

Interface Source Code

Interface source code written in C is given using this the user can write only the UDP () functions. To view the interface source code, go to

NetSim Installation path / src / Programming/ SocketUDP.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista
- Properties --> find target in Windows XP.

Sample Scenarios:

Objective - Send and **Receive** the data using **PC** to **PCCommunication** - **SocketProgramming**

How to Proceed? -Two Systems are required to perform this experiment. When one System is in the **ServerMode** other should be in the **Client Mode**. The objective can be executed in **NetSim** using the programming exercise available. In the **Programming** menu select**PC to PC Communication → Socket Programming**.

Sample Inputs - By using the **InputPanel** that is available on the left hand side a **SampleScenario** is created. The Steps involved in creating are as follows,

- **SampleMode** should be selected.
- **Select** a **Protocol** from the following,
  - **TCP**
  - **UDP**
- Under **Operation**, **Server** or **Client** should be selected.
- On the Right hand side panel there is an **InputField** "**Enter Data to be Transmitted**", where **Data** needs to be typed in. This **Data** is later sent to the **ReceiverSystem**. Here **Type** "**Hello**".
- Under **Input** there are two things,

- When **Operation** is **Client**, then the **Server'sIPAddress** (192.168.1.2) should be given in the **Server** IP **Address** field.
- When **Operation** is **Server**, then the **Server'sIPAddress** (192.168.1.1) would be automatically filled in the **Local IP Address** field.

Sample Output:

- On the Right hand side panel there is an **InputField** "**ReceivedData**", where **Data** would get displayed. "**Hello**" is the **Data** that is received from the **ClientSystem**.

TCP

- First the **Server** should **click** on the **Run** button after which the **Client** should click on the **Run** button to **Create the socket**
- **Client** should click on the **Connect** button to **establish the connection with server**.
- The **Client** should click on the **Send** button to transmit the data to the **Server.**
- The **Client** should click on the **Close** button to terminate the Connection with **Server.**
- If the **Data** is successfully transmitted then the **SentData** would be **Received** in the **ServerSystem**.

UDP

- First the **Server** should **click** on the **Run** button after which the **Client** should click on the **Run** button to **Create the socket**
- The **Client** should click on the **Send** button to transmit the data to the **Server.**
- The **Client** should click on the **Close** button to terminate the Connection with **Server.**
- If the **Data** is successfully transmitted then the **SentData** would be **Received** in the **ServerSystem**.

PC to PC Communication – Chat Application TCP

Programming Guidelines

This section guides the user to link his/her own code for PC to PC Communication – Socket programming to NetSim.

Pre – Conditions

The user program should read the input scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

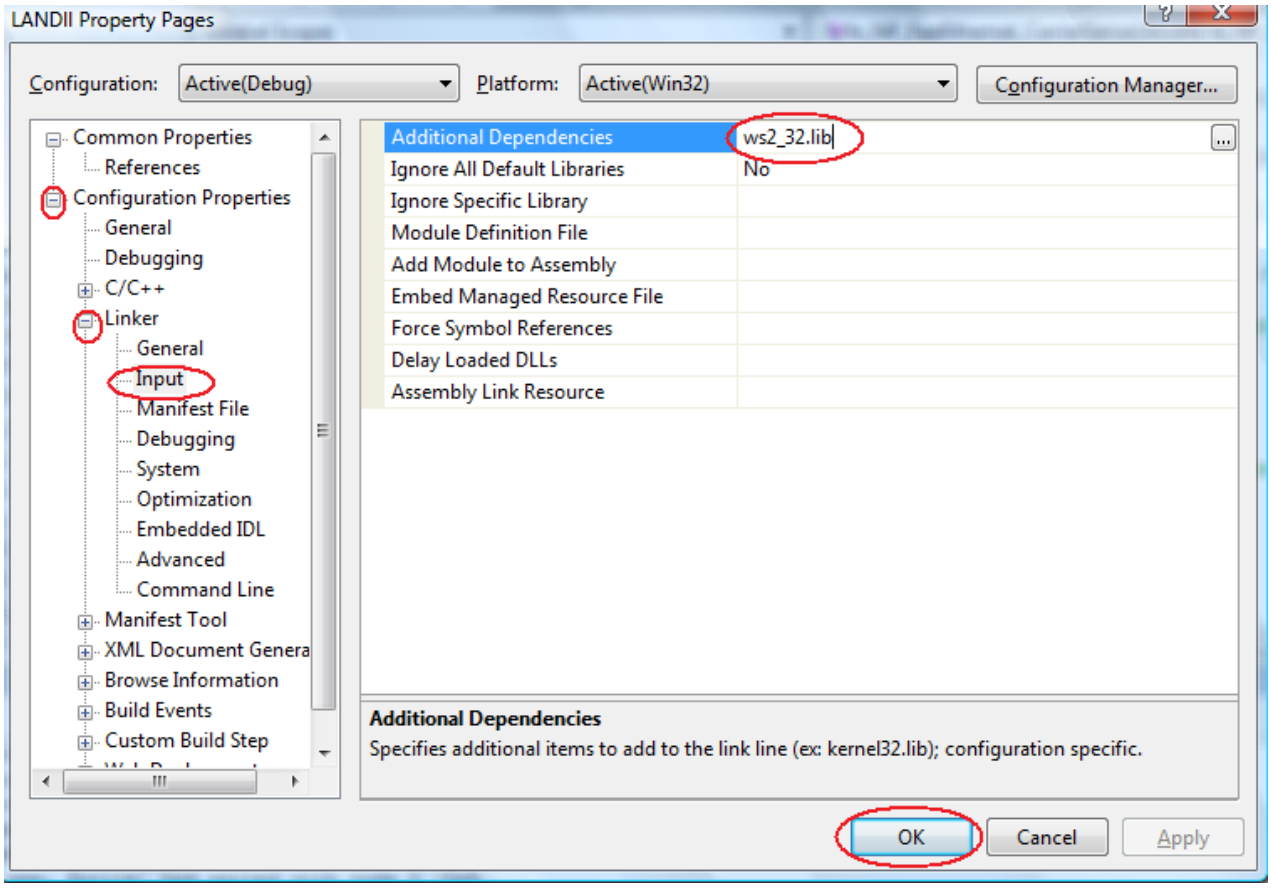
Executing the required concept and

The results of the program should be written into the output file **Output.txt**.

Coding Pre Requisites:

The prerequisite for the creation of sockets in the Windows-based C complier is they must have **winsock.h** header file in its standard header file folder. In order to use the commands of the winsock.h, there must be wsock32.lib library file present in the C complier. To link wsock32.lib, different steps have to be followed for different compilers as shown:

- For Visual Studio complier 2010 (10.0 versions), the library file linkage must be made in the following way:



a. Click **Project Menu** in the workspace.

b. Select **properties** in the pull-down menu or press Alt + F7

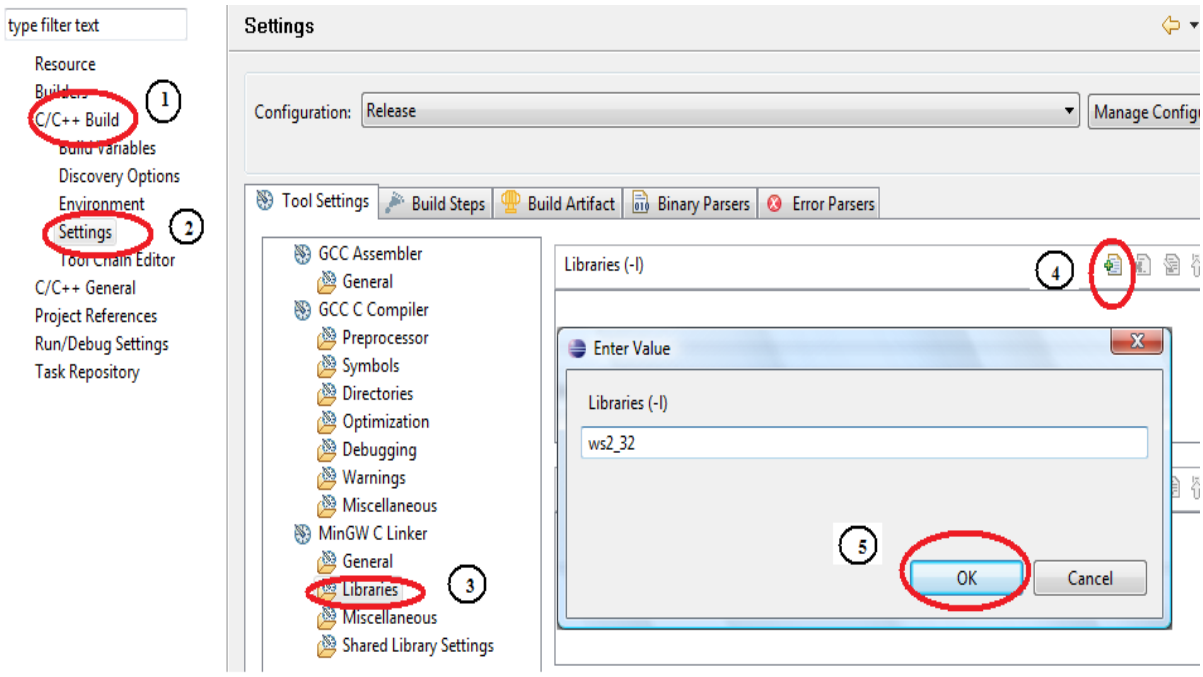
c. Then click **Linker.**

d. Select **Input**

d. Type **ws2_32.lib** file in the **Additional Dependencies** text box area.

e. Click **OK** to confirm the changes in the workspace.

- If you have an eclipse complier, the library linkage must be in the following way



a. Click **Project Menu** in the workspace.

b. Select **properties** in the pull-down menu

c. Then click C/C++ Build**.**

d. Select **Setting.**

e. Select **Libraries** under MinGW C Linker

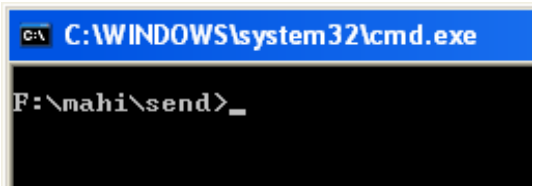
f. Press **add** button in the Libraries (-l) window

d. Type **ws2_32** file in the text box area.

e. Click **OK** to confirm the changes in the workspace.

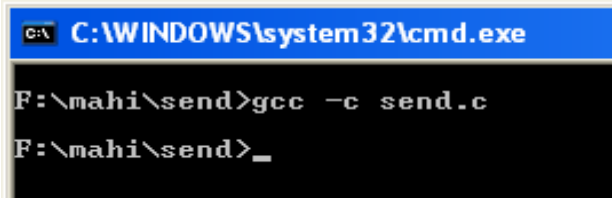
- If gcc compiler is used via command prompt:

a. Set the path to Socket program location.



b. Create output file (.o file) using below command.

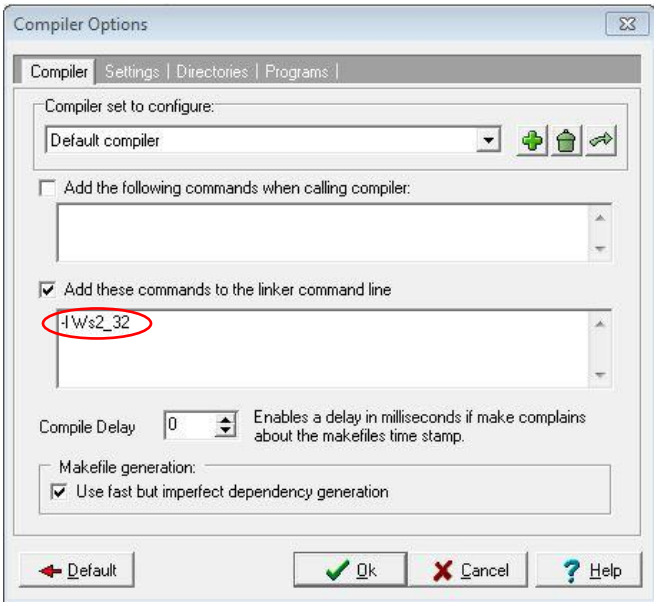
gcc -C Filename.c



c. Create exe file (.exe file) using below command

gcc -o send send.o -lws2_32

- If using Dev C++ IDE, go to Tools->compiler options and add -lws2_32 as follows before executing:



Input File Format	Output File Format
The data format in the input file is as follows, **Receive Input** Protocol=TCP Local User Name=Tetcos **Send** Protocol=TCP Number of Users=2 1=192.168.0.1 2=192.168.0.2	Write the received message Ex: Tetcos:-Hello!!!!!!!!!!!!!!

Tetcos:-Hello!!!!!!!!!!	

Interface Source Code

Interface source code written in C is given using this the user can write the TCP chat functions.

To view the interface source code, go to

NetSim Installation path / src / Programming/ ChatTCP.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista
- Properties --> find target in Windows XP.

PC to PC Communication – Chat Application UDP

Programming Guidelines

This section guides the user to link his/her own code for PC to PC Communication – Socket programming to NetSim.

Pre – Conditions`

The user program should read the input scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

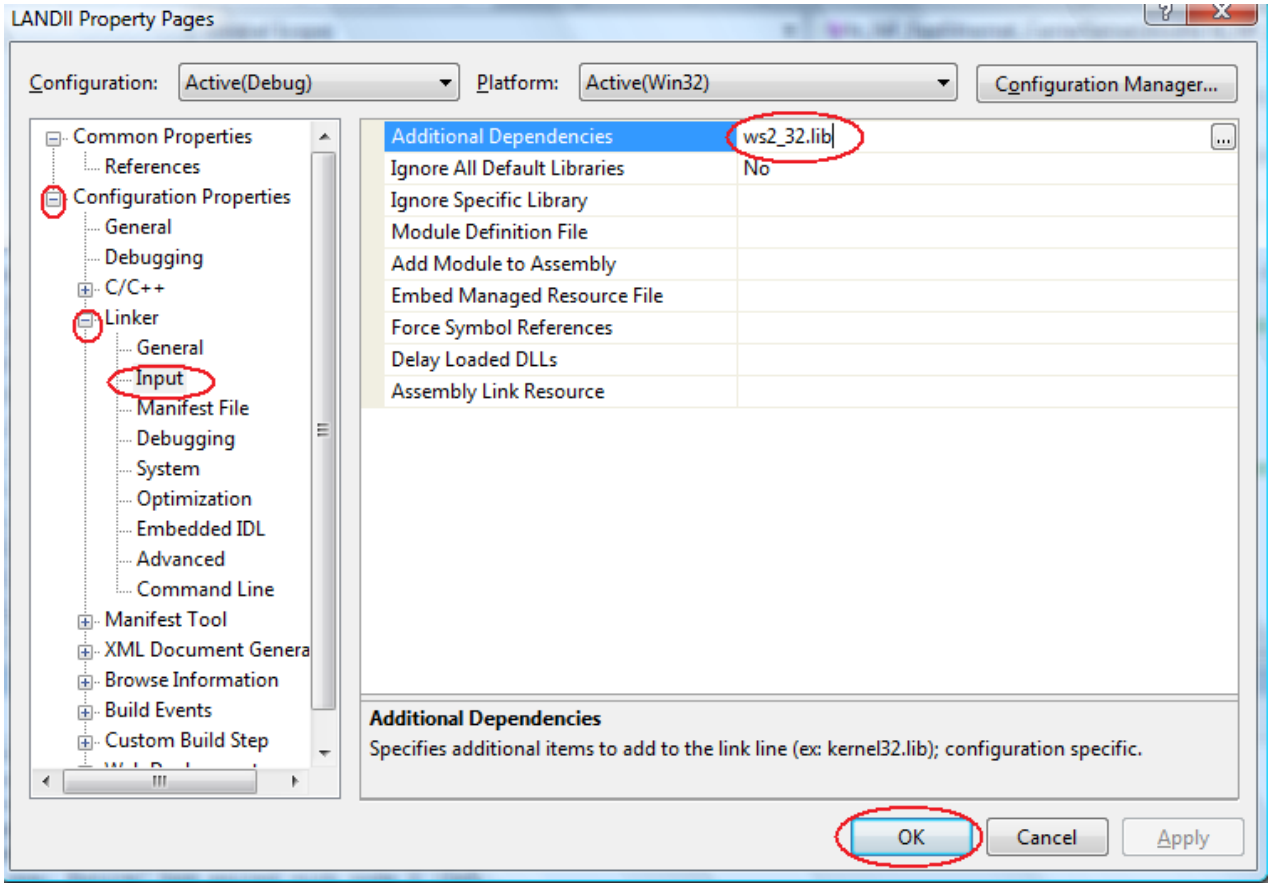
Executing the required concept and

The results of the program should be written into the output file **Output.txt**.

Coding Pre Requisites:

The prerequisite for the creation of sockets in the Windows-based C complier is they must have **winsock.h** header file in its standard header file folder. In order to use the commands of the winsock.h, there must be wsock32.lib library file present in the C complier. To link wsock32.lib, different steps have to be followed for different compilers as shown:

For Visual Studio complier 2010 (10.0 versions), the library file linkage must be made in the following way:



a. Click **Project Menu** in the workspace.

b. Select **properties** in the pull-down menu or press Alt + F7

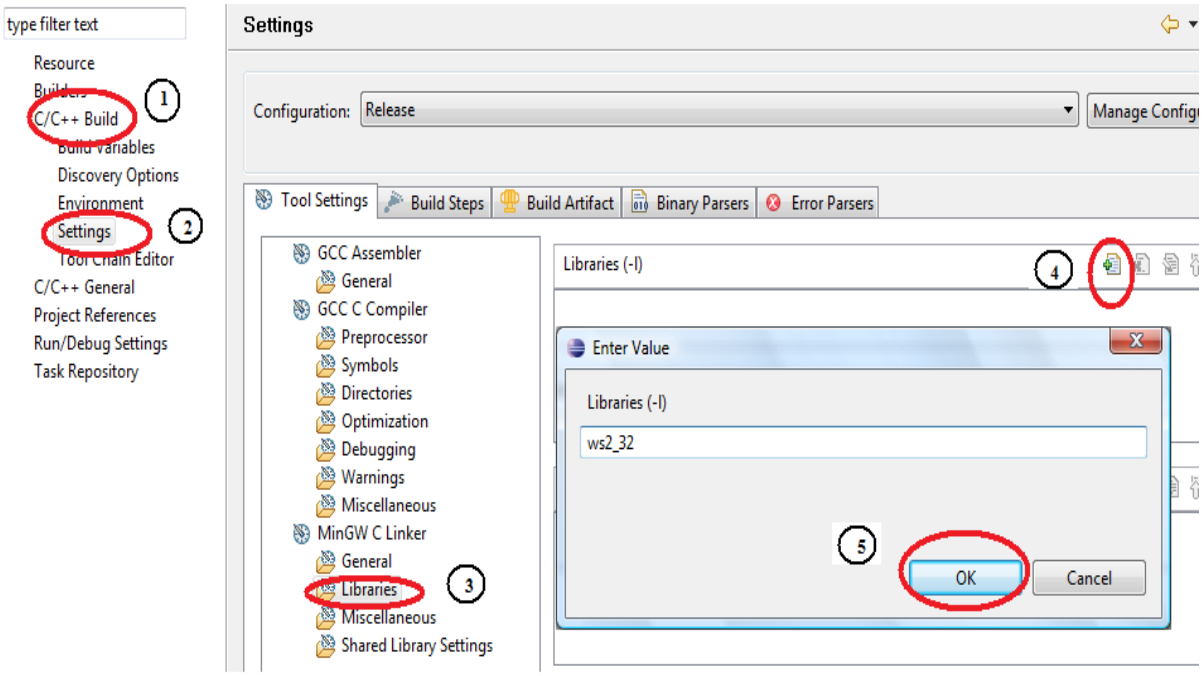
c. Then click **Linker.**

d. Select **Input**

d. Type **ws2_32.lib** file in the **Additional Dependencies** text box area.

e. Click **OK** to confirm the changes in the workspace.

- If you have an eclipse complier, the library linkage must be in the following way



a. Click **Project Menu** in the workspace.

b. Select **properties** in the pull-down menu

c. Then click C/C++ Build**.**

d. Select **Setting.**

e. Select **Libraries** under MinGW C Linker

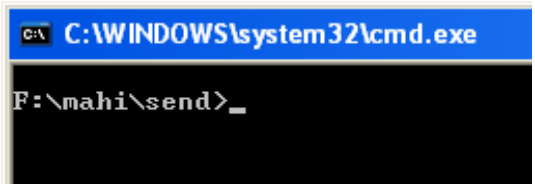
f. Press **add** button in the Libraries (-l) window

d. Type **ws2_32** file in the text box area.

e. Click **OK** to confirm the changes in the workspace.

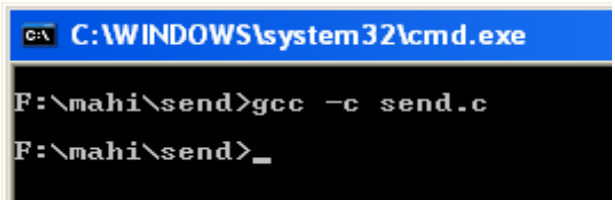
- If gcc compiler is used via command prompt:

a. Set the path to Socket program location.



b. Create output file (.o file) using below command.

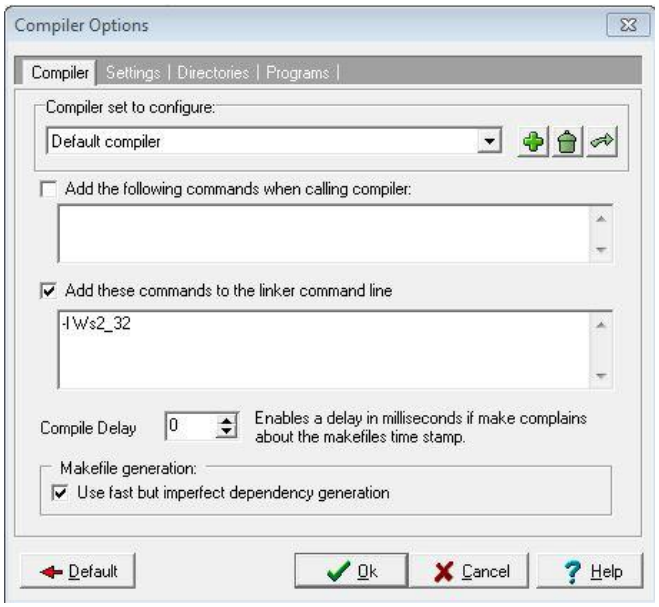
gcc -C Filename.c



c. Create exe file (.exe file) using below command

gcc -o send send.o -lws2_32

- If using Dev C++ IDE, go to Tools->compiler options and add -lws2_32 as follows before executing:



Input File Format	Output File Format
The data format in the input file is as follows,.	Write the received message Ex:

Receive Input Protocol=UDP Local User Name=Tetcos **Send** Protocol=TCP Number of Users=2 1=192.168.0.1 2=192.168.0.2 Tetcos:-Hello!!!!!!!!!!	192.168.0.2:- Requesting for the connection

Interface Source Code

Interface source code written in C is given using this the user can write UDP Chat functions. To view the interface source code, go to

NetSim Installation path / src / Programming/ ChatUDP.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista
- Properties --> find target in Windows XP.

Sample Scenarios:

Objective - Send and **Receive** the data using **PC** to **PCCommunication** – **Chat Application**

How to Proceed? -Two to Ten Systems are required to perform this experiment.

The objective can be executed in **NetSim** using the programming exercise available. In the **Programming** menu select**PC to PC Communication → Chat Application**.

Sample Inputs - By using the **InputPanel** that is available on the left hand side a **SampleScenario** is created. The Steps involved in creating are as follows,

- **SampleMode** should be selected.
- **Select** a **Protocol** from the following,
    - **TCP**

- **UDP**

- On the Right hand side panel there is an **InputField** "**Send Data**", where **Data** needs to be typed in. This **Data** is later sent to the **ReceiverSystem**. Here **Type** "**Hello**".
- On the Right hand side panel there is "**Receive Data**", where data is received from user. Ex. **"Hello"**
- Under **Input** there are two things,
  - Number of users must be select between one to ten
  - Enter the IP address of the data where needs to be in all the column in the input pane.

Sample Output:

- On the Right hand side panel there is an **InputField** "**ReceivedData**", where **Data** would get displayed. "**Hello**" is the **Data** that is received from the **users**.

TCP

- Click **Run** button to **create the socket** and start Receive Data from the users.
- Click **Send** button to transmit the data to the users**.**
- Click **Refresh** button to terminate the connection.

UDP

- Click **Run** button to **create the socket** and start Receive Data from the users.
- Click **Send** button to transmit the data to the users**.**

Scheduling - First In First Out (FIFO)

Programming Guidelines

This section guides the user to link his/her own code for Scheduling - FIFO to NetSim.

Pre - Conditions

The program should read the data from '**Input.txt**' file that is available in the temporary directory.

The program should write the output data to '**Output.txt'** file that is available in the temporary directory.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%" → NetSim → "Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and, the results of the program should be written into the output file **Output.txt**.

Note: The naming of the input and the output file must be same as the text displayed in the Methodology screen

File Format

Input File Format	Output File Format
Output_Link_Capacity=100 Node's_Priority=8,7,6,5,4,3,2,1, Node's_Capacity=0,0,500,0,0,0,0,0,	Output.txt file format is as follows, Seconds>no of bits transmitted in node 1>Transmission sequence of node 1>... no of bits transmitted in node 8>Transmission sequence of node 8> **Sample Output File** 1>0>0>0>0>100>1>0>0>0>0>0>0>0>0>0>0> 2>0>0>0>0>100>1>0>0>0>0>0>0>0>0>0>0> 3>0>0>0>0>100>1>0>0>0>0>0>0>0>0>0>0> 4>0>0>0>0>100>1>0>0>0>0>0>0>0>0>0>0> 5>0>0>0>0>100>1>0>0>0>0>0>0>0>0>0>0>

Interface Source Code

Interface source code written in C is given using this the user can write only the Scheduling inside the function fnFIFO() using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ FIFO.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista
- Properties --> find target in Windows XP.

Sample Scenarios:

Objective - The objective is to allocate the output capacity among the various input capacities using **First In First Out (FIFO) Scheduling** algorithm.

How to Proceed? - The objective can be executed in **NetSim** using the programming exercise available. In the **Programming** menu select**Scheduling**.

Sample Input - By using the **InputPanel** that is available on the left hand side a **SampleScenario** is created. The Steps involved in creating are as follows,

- **SampleMode** should be selected.
- **Algorithm** should be selected as **FirstIn First Out (FIFO).**
- Under **Input** field,
    0. **EnterOutputLinkCapacity** which is in **MBPS**. The options that are available for selecting are 100, 200, 300, …… 1000 MBPS. Select **400 MBPS**.
    1. **EnterBits** / **Node** for one second (limit 0 - 9999) in the fields available. i.e. Node1 - "100" , Node2 - "100", …….., Node8 - "100".
    - The **Priority** for the **Node** needs to be given. Click to Select or Deselect the Priority of a Node. In this **Sample** the **Priorities** are as follows,

Node Number	Priority Given
1	7
2	6
3	8
4	3
5	2
6	1
7	5
8	4

- **Click** on the **Run** button.

Output

Note: The Transmission of the data takes place in the following sequence. The Bits / Node are represented in different colors. These colors indicate the order in which the transmission takes place.

The Transmission Sequence → **1** **2** **3** **4** **5** **6** **7** **8.**

Seconds	1	2
Node1	0	100
Node2	0	100
Node3	100	0
Node4	0	100
Node5	0	100
Node6	100	0
Node7	100	0
Node8	100	0

Scheduling - Max - Min Fair (MMF)

Programming Guidelines

This section guides the user to link his/her own code for Scheduling - MMF to NetSim.

Pre - Conditions

The program should read the data from '**Input.txt**' file that is available in the application path.

The program should write the output data to '**Output.txt'** file that is available in the application path.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%" → NetSim → "Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and, the results of the program should be written into the output file **Output.txt**.

Note: The naming of the input and the output file must be same as the text displayed in the Methodology screen

File Format

Input File Format	Output File Format
Output_Link_Capacity =100 Node's_Capacity=0,0, 300,0,0,0,0,0,	'**Output.txt**' file format is as follows. 1st second >Output rate of node 1 >Priority of Node 1>……..>Output rate of node 8.Priority of Node8 2nd second >Output rate of node 1 >Priority of Node 1>……..>Output rate of node 8.Priority of Node8 Nth second >Output rate of node 1 >Priority of Node 1>……..>Output rate of node 8.Priority of Node8 **Sample File data** 1>0.00>0>0.00>0>100.00>1>0.00>0>0.00>0>0.00>0>0.00>0>0.00>0> 2>0.00>0>0.00>0>100.00>1>0.00>0>0.00>0>0.00>0>0.00>0>0.00>0> 3>0.00>0>0.00>0>100.00>1>0.00>0>0.00>0>0.00>0>0.00>0>0.00>0>

Interface Source Code

Interface source code written in C is given using this the user can write only the Scheduling inside the function fnMMFA () using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ MMF.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista
- Properties --> find target in Windows XP.

Sample Scenarios:

Objective - The objective is to allocate the output capacity among the various input capacities using **Max-Min Fare(MMF) Scheduling** algorithm.

How to Proceed? - The objective can be executed in **NetSim** using the programming exercise available. In the **Programming** menu select**Scheduling**.

Sample Input - By using the **InputPanel** that is available on the left hand side a **SampleScenario** is created. The Steps involved in creating are as follows,

- **SampleMode** should be selected.
- **Algorithm** should be selected as **Max - Min Fair (MMF).**
- Under **Input** field,
    0. **EnterOutputLinkCapacity** which is in **MBPS**. The options that are available for selecting are 100, 200, 300, …… 1000 MBPS. **Select100MBPS**.
    1. **EnterBits / Node** for one second (limit 0 - 9999) in the fields available.
    i.e. Node1 - "100" , Node2 - "100", ……..., Node8 - "100".
- **Click** on the **Run** button.

Output

The Transmission of the data takes place in the following sequence. The **Bits / Node** are represented in different colors. These colors indicate the order in which the transmission takes place.

The Transmission Sequence → 1 2 3 4 5 6 7 8.

Seconds	1	2	3	4	5	6	7	8
Node1	12.50	12.50	12.50	12.50	12.50	12.50	12.50	12.50
Node2	12.50	12.50	12.50	12.50	12.50	12.50	12.50	12.50
Node3	12.50	12.50	12.50	12.50	12.50	12.50	12.50	12.50
Node4	12.50	12.50	12.50	12.50	12.50	12.50	12.50	12.50
Node5	12.50	12.50	12.50	12.50	12.50	12.50	12.50	12.50
Node6	12.50	12.50	12.50	12.50	12.50	12.50	12.50	12.50
Node7	12.50	12.50	12.50	12.50	12.50	12.50	12.50	12.50
Node8	12.50	12.50	12.50	12.50	12.50	12.50	12.50	12.50

Shortest Path - Floyd's

Programming Guidelines

This section guides the user to link his/her own code for shortest path – Floyd's to NetSim.

Pre - Conditions

The program should read the data from 'Input.txt' file that is available in the temporary directory.

The program should write the output data to 'Output.txt' file that is available in the temporary directory.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%" → NetSim → "Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file Input.txt.

Executing the required concept and, the results of the program should be written into the output file Output.txt.

Input file format	**Output file format**
Algorithm=Floyd's No_of_Router=3 Distance: 999>5>1> 5>999>10> 1>10>999> Source_Router=1 Destination_Router=3 ***Note:*** '>' is the delimiter symbol, which is used to separate each input.	Consider the source node is 1 Distance>0>1>2>1>2>1> Path>0>1>2>1>4>1> The first line has the all the distance values from the source node to all the other connected nodes. The second line has the path values from the source node to all the other nodes. The first line contains the values stored in the distance array (single dimensional array) used in the algorithm. The second line contains the values stored in the path array (single dimensional array) used in the algorithm. Sample Output: 1>2>3> 1>2> 1>3> 1> >>> path_FS> 0>0>0> 0>0>1> 0>1>0> distance_FS> 0>5>1> 5>0>6> 1>6>0> *Note:* The string Distance and Path in the file is compulsory. The output must be stored in the same format

Interface Source Code

Interface source code written in C is given using this the user can write only the Scheduling inside the function fnFloyd() using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ Floyds.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista
- Properties --> find target in Windows XP.

Sample Scenarios:

Objective - The objective is to find the **ShortestPath** between the two **Routers** using the **Floyds** algorithm.

How to Proceed? - The objective can be executed in **NetSim** using the **Programming** exercise available. Under **ProgrammingMenu** select **ShortestPath**.

Sample Input - By using the **Input** Panel that is available on the left hand side a **SampleScenario** is created. The **Steps** involved in creating are as follows,

- **SampleMode** should be selected.
- **Select** the **Algorithm** as **Floyd's**.
- **Number** of **Routers** needs to be selected. **Minimum** of **3Routers** and **Maximum** of **8Routers** can be selected. Let us consider 4 **Routers** are selected.
- **SourceRouter** needs to be given. Let **Router** 4 be the **SourceRouter**.
- **DestinationRouter** also needs to be given. Let **Router** 1 be the **DestinationRouter**.
- **Click** on 2 **Routers** to give the **distance** between those 2 **Routers**. A **bluearrow** would be pointing from the **SourceNode** to the **DestinationNode**. The **Distances** between the **Routers** that needs to be given are as follows,
    1. Router 4 and Router 1 → 67 Km,
    2. Router 4 and Router 3 → 46 Km,
    3. Router 3 and Router 1 → 18 Km,

4. Router 3 and Router 2 → 56 Km, and
5. Router 1 and Router 2 → 70 Km.

- **Click** on **Run** button to view the output in the **RoutingTable**.

Output - The **Output** for the above **Scenario** is as follows,

- **Click** the **Run** button to view program output.
- The **Output** is obtained in a **RoutingTable**. All possible **Routes** and the corresponding **Distance** are obtained. Below is a **RoutingTable** for the above Inputs,

Routing Table	
Route	**Distance[KM]**
4>1>	67
4>3>1>	64
4>3>2>	102
4>3>	46
4>	0

- The **ShortestPath** would be highlighted in **greencolor**.
- **Click** on **RefreshButton** to **Refresh** the screen and create fresh **Scenarios**.

Shortest Path - Link State

Programming Guidelines

This section guides the user to link his/her own code for shortest path - Link State Routing to NetSim.

Pre - Conditions

The program should read the data from '**Input.txt**' file that is available in the temporary directory.

The program should write the output data to '**Output.txt'** file that is available in the temporary directory.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%" → NetSim → "Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and, the results of the program should be written into the output file **Output.txt**.

Input file format	**Output file format**
Input.txt contains the distance matrix given by the user, along with the source node, destination node and the total number of nodes. Here, we have the input as a SYMMETRIC matrix. The format of input.txt is, for example, Algorithm=Link_State No_of_Router=3 Distance: 999>6>5> 6>999>7> 5>7>999> Source_Router=1 Destination_Router=2 *Note:* '>' is the delimiter symbol, which is used to separate each input.	Consider the source node is 1 Distance>0>1>2>1>2>1> Path>0>1>2>1>4>1> The first line has the all the distance values from the source node to all the other connected nodes. The second line has the path values from the source node to all the other nodes. The first line contains the values stored in the distance array (single dimensional array) used in the algorithm. The second line contains the values stored in the path array (single dimensional array) used in the algorithm. Sample Output: 1>2> 1>3>2> 1>3> 1> >>> Distance>0>6>5> Path>1000>1>1> *Note:* The string Distance and Path in the file is compulsory. The output must be stored in the same format.

Interface Source Code

Interface source code written in C is given using this the user can write only the Scheduling inside the function fnDijkstra() using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ LinkState.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista
- Properties --> find target in Windows XP.

Sample Scenarios:

Objective - The objective is to find the **Shortest Path** between the two **routers** using the **Link State** algorithm.

How to Proceed? - Under **ProgrammingMenu** select **ShortestPath**.

Sample Input - By using the **InputPanel** that is available on the left hand side a **SampleScenario** is created. The Steps involved in creating are as follows,

- **SampleMode** should be selected.
- **Select** the **Algorithm** as **RunedState**.
- **Number** of **Routers** needs to be selected. **Minimum** of **3Routers** and **Maximum** of **8Routers** can be selected. Select 5 **Routers**.
- **SourceRouter** needs to be given. Let **Router** 5 be the **SourceRouter**.
- **DestinationRouter** also needs to be given. Let **Router** 3 be the **DestinationRouter**.
- **Click** on 2 **Routers** to give the distance between those 2 **Routers**. A blue Line would be pointing from the **SourceNode** to the **DestinationNode**. The **Distances** between the **Routers** that needs to be given are as follows,
    1. Router 1 and Router 2 → 35,
    2. Router 2 and Router 3 → 30,
    3. Router 3 and Router 4 → 60,

4. Router 4 and Router 1 → 20,
5. Router 4 and Router 5 → 67, and
6. Router 5 and Router 2 → 58.

- **Click** on **Run** button to view the output in the **RoutingTable**.

Output - The **Output** for the above **Scenario** is as follows,

- **Click** the **Run** button to view program output.
- The **Output** is obtained in a **RoutingTable**. All possible **Routes** and the corresponding **Distance** are obtained. Below is a **RoutingTable** for the above Inputs,

Routing Table

Route	Distance[KM]
5>2>1>4>3>	173
5>2>1>4>	113
5>2>1>	93
5>2>3>	88
5>2>	58
5>4>1>2>3>	152
5>4>1>2>	122
5>4>1>	87
5>4>3>	127
5>4>	67
5>	0

- The **ShortestPath** would be highlighted in **Green** color in the **RoutingTable**.
- **Click** on **RefreshButton** to **Refresh** the screen and create fresh **Scenarios**.

Sliding Window Protocol - Go Back N

Programming Guidelines

This section guides the user to link his/her own code for Go Back N to NetSim.

Pre - Conditions

The user program should read the input scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%" → NetSim → "Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and

The results of the program should be written into the output file **Output.txt**.

Window Size:

The **Window Size** for **Go Back N** is **7**

Input File Format	Output File Format
Algorithm=Go_Back_N Data_File=C:\Users\P.Sathishkumar\Documents\1 Th.txt> Bit_Error_Rate=6 Sequence_Number=3 Window_Size=7	Value1>Value2>Value3>Value4> **Types:** There are five types of formats of writing in the output file. Each format is written in specific condition, the types are explained below. The condition of writing the types is explained in the algorithm. **Type1:** Value1 - "CNT", Value 2 -output of the slidingcount function Value 3 - "FRAMES" Value 4 - "TRANSMIT" **Type2:**

	Value1 - "DT", Value 2 -Frame number, Value 3 - Frame's Source address, Value 4 - Frame's Destination address. **Type3**: Value 1 - "EV", Value 2 - Output of the intro_error function, Value 3 - Frame' Source address, Value 4 - Frame's Destination address. **Type4**: Value 1 - "ACK", Value 2 - "POS", Value 3 - Acknowledgement frame's Source Address, Value 4 - Acknowledgement frame's Destination Address. **Type5:** Value 1 - "DEL" Value 2 - count of frames being deleted Value 3 - "FRAME" Value 4 - "DELETED" **Note:** The above convention to write into the `'Output.Txt'` is**mandatory**. Values in Quotes`""` to be written into file `'Output.Txt'` as it is including Case. DT>Frame No>node1>node2> (DT denotes Data from node 1 to node 2) EV>Error Flag>node1>node2> (EV denotes Error Value - i.e If the above frame has error then set the error flag as 1 else set the flag as 0) ACK>POS>node2>node1> (Acknowledgement for that above frame received is sent to node 2 to node 1) **Ex:** CNT>1>FRAMES>TRANSMIT> DT>1>node1>node2> EV>0>node1>node2> ACK>POS>node1>node2>

Interface Source Code

Interface source code written in C is given using this the user can write only the Sliding Window Protocols - Go Back N inside the function GoBackN () using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ GobackN_SW.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista
- Properties --> find target in Windows XP.

Sample Scenarios:

Objective - Send and receive the data using **Sliding Window Protocol - Go BackN**.

How to Proceed? - The objective can be executed in **NetSim** using the programming exercise available. Under **ProgrammingMenu** select **Sliding WindowProtocol→Go Back N**.

Sample Input - By using the **InputPanel** that is available on the left hand side a **SampleScenario** is created. The Steps involved in creating are as follows,

- **SampleMode** should be selected.
- **SelectGoBackN** as **Algorithm**.
- **Create** a "**Datafile** (**.txt**)" and save it on the disk.
- **Select** the path of the above created "**Datafile** (**.txt**)" which could be **Maximum** of **100000bytes**.
- **SelectBit Error Rate** (**BER**) as "**10^-5**" from the drop down menu.
- **SelectSequencenumber** (**Bits**) as "**3**" from the drop down menu.
- A default value is entered for **WindowSize** as "**7**".

1. Then **Run** button need to be clicked.

Output - **Output** for the above **Sample** is as follows,

2. An **OutputTable** is obtained when **Run** button is clicked. In the **OutputTable**, **Transmittedframe** (**Data** and **Acknowledgement**) and their corresponding Counts is obtained. The **TotalCount** is also obtained. The table is given below,

Output

Total data frames to be transmitted - 24

Transmitted frame	Count
Data	30
Acknowledgement	5
	Total = 35

Note - The "Total data frames to be transmitted" and "Total count" in the **Output** table depends on size of the "**.txt**" file.

3. The details of the **DataFrames** flowing from **Node1** to **Node 2** are obtained on the right hand side panel. Below are the details that is obtained in the tool,
    - Data Frame 1 is flowing from Node1 to Node2 with no error.
    - Data Frame 2 is flowing from Node1 to Node2 with no error.
    - Data Frame 3 is flowing from Node1 to Node2 with no error.
    - Data Frame 4 is flowing from Node1 to Node2 with error.
    - Data Frame 5 is flowing from Node1 to Node2 with no error.
    - Data Frame 6 is flowing from Node1 to Node2 with no error.
    - Data Frame 7 is flowing from Node1 to Node2 with error.
    - Acknowledgement from Node2 to Node1.
    - Data Frame 4 is flowing from Node1 to Node2 with no error.
    - Data Frame 5 is flowing from Node1 to Node2 with no error.
    - Data Frame 6 is flowing from Node1 to Node2 with no error.
    - Data Frame 7 is flowing from Node1 to Node2 with no error.
    - Data Frame 8 is flowing from Node1 to Node2 with no error.
    - Data Frame 9 is flowing from Node1 to Node2 with no error.
    - Data Frame 10 is flowing from Node1 to Node2 with no error.
    - Acknowledgement from Node2 to Node1.
    - Data Frame 11 is flowing from Node1 to Node2 with no error.
    - Data Frame 12 is flowing from Node1 to Node2 with no error.

- Data Frame 13 is flowing from Node1 to Node2 with no error.
- Data Frame 14 is flowing from Node1 to Node2 with no error.
- Data Frame 15 is flowing from Node1 to Node2 with no error.
- Data Frame 16 is flowing from Node1 to Node2 with no error.
- Data Frame 17 is flowing from Node1 to Node2 with no error.
- Acknowledgement from Node2 to Node1.
- Data Frame 18 is flowing from Node1 to Node2 with no error.
- Data Frame 19 is flowing from Node1 to Node2 with no error.
- Data Frame 20 is flowing from Node1 to Node2 with no error.
- Data Frame 21 is flowing from Node1 to Node2 with no error.
- Data Frame 22 is flowing from Node1 to Node2 with no error.
- Data Frame 23 is flowing from Node1 to Node2 with error.
- Data Frame 24 is flowing from Node1 to Node2 with no error.
- Acknowledgement from Node2 to Node1.
- Data Frame 23 is flowing from Node1 to Node2 with no error.
- Data Frame 24 is flowing from Node1 to Node2 with no error.
- Acknowledgement from Node2 to Node1.

- Once the sample experiment is done, then **Refresh** button can be clicked to create new samples.

Sliding Window Protocol - Selective Repeat

Programming Guidelines

This section guides the user to link his/her own code for Selective Repeat to NetSim.

Pre - Conditions

The user program should read the input scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%" → NetSim → "Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and

The results of the program should be written into the output file **Output.txt**.

Window Size:

The **Window Size** for Selective Repeat is **7**

Input File Format	Output File Format
Algorithm=Selective_Repeat Data_File=C:\Users\P.Sathishkumar \Documents\1 Th.txt> Bit_Error_Rate=5 Sequence_Number=3 Window_Size=4	Value1>Value2>Value3>Value4> **Types:** There are five types of formats of writing in the output file. Each format is written in specific condition, the types are explained below. The condition of writing the types is explained in the algorithm. **Type1:** Value1 - "CNT", Value 2 - output of the slidingcount function Value 3 - "FRAMES" Value 4 - "TRANSMIT" **Type2:** Value1 - "DT", Value 2 - Frame number, Value 3 - Frame's Source address, Value 4 - Frame's Destination address. **Type3**: Value 1 - "EV", Value 2 - Output of the intro_error function, Value 3 - Frame' Source address, Value 4 - Frame's Destination address. **Type4**:

	Value 1 - "ACK", Value 2 - "POS", Value 3 - Acknowledgement frame's Source Address, Value 4 - Acknowledgement frame's Destination Address. **Type5:** Value 1 - "DEL" Value 2 - count of frames being deleted Value 3 - "FRAME" Value 4 - "DELETED" **Note:** The above convention to write into the `'Output.Txt'` is**mandatory**. Values in Quotes`""` to be written into file `'Output.Txt'` as it is including Case. DT>Frame No>node1>node2> (DT denotes Data from node 1 to node 2) EV>Error Flag>node1>node2> (EV denotes Error Value - i.e If the above frame has error then set the error flag as 1 else set the flag as 0) ACK>POS>node2>node1> (Acknowledgement for that above frame received is sent to node 2 to node 1) **Ex:** CNT>1>FRAMES>TRANSMIT> DT>1>node1>node2> EV>0>node1>node2> ACK>POS>node1>node2>

Interface Source Code

Interface source code written in C is given using this the user can write only the Sliding Window Protocol - Selective Repeat inside the function SelectiveRepeat () using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ SelectiveRepeat_SW.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista
- Properties --> find target in Windows XP.

Sample Scenarios:

Objective - Send and receive the data using **Sliding Window Protocol - Selective Repeat**.

How to Proceed? - The objective can be executed in **NetSim** using the programming exercise available. Under **Programming Menu** select **Sliding WindowProtocol→Selective Repeat**.

Sample Input - By using the **Input Panel** that is available on the left hand side a **Sample Scenario** is created. The Steps involved in creating are as follows,

- **Sample Mode** should be selected.
- **Select Selective Repeat** as **Algorithm**.
- **Create** a "**Data file** (**.txt**)" and save it on the disk.
- **Select** the path of the above created "**Data file** (**.txt**)" which could be **Maximum** of **100000 bytes**.
- **Select Bit Error Rate** (**BER**) as "**10^-5**" from the drop down menu.
- **Select Sequence number** (**Bits**) as "**3**" from the drop down menu.
- A default value is entered for **Window Size** as "**4**".

1. Then **Run** button need to be clicked.

Output - **Output** for the above **Sample** is as follows,

2. An **Output Table** is obtained when **Run** button is clicked. In the **Output Table**, **Transmitted frame** (**Data** and **Acknowledgement**) and their corresponding **Counts** is obtained. The **Total Count** is also obtained. The table is given below,

Output

Total data frames to be transmitted - 24

Transmitted frame	Count
Data	26
Acknowledgement	7

Total = 33

Note - The "Total data frames to be transmitted" and "Total count" in the **Output** table depends on size of the "**.txt**" file.

3. The details of the **Data Frames** flowing from **Node 1** to **Node 2** are obtained on the right hand side panel. Below are the details that is obtained in the tool,
    - Data Frame 1 is flowing from Node1 to Node2 with no error
    - Data Frame 2 is flowing from Node1 to Node2 with no error
    - Data Frame 3 is flowing from Node1 to Node2 with no error
    - Data Frame 4 is flowing from Node1 to Node2 with error
    - Acknowledgement from Node2 to Node1
    - Data Frame 4 is flowing from Node1 to Node2 with no error
    - Data Frame 5 is flowing from Node1 to Node2 with no error
    - Data Frame 6 is flowing from Node1 to Node2 with error
    - Data Frame 7 is flowing from Node1 to Node2 with no error
    - Acknowledgement from Node2 to Node1
    - Data Frame 6 is flowing from Node1 to Node2 with no error
    - Data Frame 8 is flowing from Node1 to Node2 with no error
    - Data Frame 9 is flowing from Node1 to Node2 with no error
    - Data Frame 10 is flowing from Node1 to Node2 with no error
    - Acknowledgement from Node2 to Node1
    - Data Frame 11 is flowing from Node1 to Node2 with no error
    - Data Frame 12 is flowing from Node1 to Node2 with no error
    - Data Frame 13 is flowing from Node1 to Node2 with no error
    - Data Frame 14 is flowing from Node1 to Node2 with no error
    - Acknowledgement from Node2 to Node1
    - Data Frame 15 is flowing from Node1 to Node2 with no error
    - Data Frame 16 is flowing from Node1 to Node2 with no error
    - Data Frame 17 is flowing from Node1 to Node2 with no error
    - Data Frame 18 is flowing from Node1 to Node2 with no error
    - Acknowledgement from Node2 to Node1
    - Data Frame 19 is flowing from Node1 to Node2 with no error

- Data Frame 20 is flowing from Node1 to Node2 with no error
- Data Frame 21 is flowing from Node1 to Node2 with no error
- Data Frame 22 is flowing from Node1 to Node2 with no error
- Acknowledgement from Node2 to Node1
- Data Frame 23 is flowing from Node1 to Node2 with no error
- Data Frame 24 is flowing from Node1 to Node2 with no error
- Acknowledgement from Node2 to Node1

- Once the sample experiment is done, then **Refresh** button can be clicked to create new samples.

Sorting Technique - Bubble Sort

Programming Guidelines

This section guides the user to link his/her own code for Sorting Algorithm to NetSim.

Pre - Condition

User written program should read the value from the '**Input.txt**' in the temporary directory which is having input from the GUI at runtime

The output should be stored in '**Output.txt**' in the temporary directory for display.

User written program should return an integer value.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and, The results of the program should be written into the output file **Output.txt**.

Note: The naming of the input and the output file must be same as the text displayed in the Methodology screen

File Format

Input File Format	Output File Format
Ascending Order Sorting_Type=Bubble Sorting_Order=Ascending Total_Number=3 Number_to_Sort=5,4,3 **Descending Order** Sorting_Type=Bubble Sorting_Order=Descending Total_Number=3 Number_to_Sort=4,5,6	**type>index 1>index 2>** type 0 : specifies Positioning the data value index that ha type 1 : specifies the two index following it are being compared type 2 : specifies the two index following it are being swapped The following types are used in Quick Sort : type 3 : specifies the two index, where index 1 value is copied to index 2 value. type 4 : specifies the two index, where Position data index value of index1 is copied to the index 2 value. index 1 : index 1 is the index of array that points to the first data that is being swapped or compared. index 2 : index 2 is the index of array that points to the second data that is being swapped or compared. **Sample Output:** **Ascending Order:** 1>0>1> 2>0>1> 1>1>2> 2>1>2> 1>0>1> 2>0>1> **Descending Order:** 1>0>1> 2>0>1> 1>1>2> 2>1>2> 1>0>1> 2>0>1>

Interface Source Code

Interface source code written in C is given using this the user can write only the Sorting Algorithm inside the function fnBubblesort () using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ BubbleSort.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista
- Properties --> find target in Windows XP.

Sample Scenarios:

Objective - To study the working of Sorting Techniques.

How to Proceed? -

The objective can be executed in NetSim using the programming exercise available, under programming user has to select Sorting Techniques.

Sample Inputs - In the Input panel the following steps need to be done,

- **SampleMode** should be **selected**.
- Select the Bubble Sort as Sorting Type.
- Select the Sorting Order as either Ascending or Descending.
- Select the total number that has to be sorted. The values available are from **3** to **25**.
- Enter the Number Value in the field provided. The value entered should be within the range of **1** to **9999**.
- Then **Run** button need to be **clicked**. **Refresh** button can be used if new Inputs have to be given.

Output - The following steps are under gone internally,

- According to the Sorting Type selected the Output would vary.

- Number of Comparison would be obtained.
- Number of Swapping would be obtained.
- A table with the values tabulated would be obtained.
- Once the sample experiment is done, then **Refresh** button can be **clicked** to create New Samples.

Sorting Technique - Insertion Sort

Programming Guidelines

This section guides the user to link his/her own code for Sorting Algorithm to NetSim.

Pre - Condition

User written program should read the value from the '**Input.txt**' in the temporary directory which is having input from the GUI at runtime

The output should be stored in '**Output.txt**' in the temporary directory for display.

User written program should return an integer value.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and, The results of the program should be written into the output file **Output.txt**.

Note: The naming of the input and the output file must be same as the text displayed in the Methodology screen

File Format

Input File Format	Output File Format
Ascending Order Sorting_Type=Insertion Sorting_Order=Ascending Total_Number=3 Number_to_Sort=9,8,7 **Descending Order** Sorting_Type=Insertion Sorting_Order=Descending Total_Number=3 Number_to_Sort=1,9,3	**type>index 1>index 2>** type 0 : specifies Positioning the data value index that ha type 1 : specifies the two index following it are being compared type 2 : specifies the two index following it are being swapped The following types are used in Quick Sort : type 3 : specifies the two index, where index 1 value is copied to index 2 value. type 4 : specifies the two index, where Position data index value of index1 is copied to the index 2 value. index 1 : index 1 is the index of array that points to the first data that is being swapped or compared. index 2 : index 2 is the index of array that points to the second data that is being swapped or compared. Sample Output: **Ascending Order:** 1>0>1> 2>0>1> 1>1>2> 2>1>2> 1>0>1> 2>0>1> **Descending Order:** 1>0>1> 2>0>1> 1>1>2> 2>1>2> 1>0>1>

Interface Source Code

Interface source code written in C is given using this the user can write only the Sorting Algorithm inside the function fnInsertsort () using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ InsertionSort.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista
- Properties --> find target in Windows XP.

Sample Project:

Objective - To study the working of Sorting Techniques.

How to Proceed? -

The objective can be executed in NetSim using the programming exercise available, under programming user has to select Sorting Techniques.

Sample Inputs - In the Input panel the following steps need to be done,

- **SampleMode** should be **selected**.
- Select the Insertion Sort as Sorting Type.
- Select the Sorting Order as either Ascending or Descending.
- Select the total number that has to be sorted. The values available are from **3** to **25**.
- Enter the Number Value in the field provided. The value entered should be within the range of **1** to **9999**.
- Then **Run** button need to be **clicked**. **Refresh** button can be used if new Inputs have to be given.

Output - The following steps are under gone internally,

- According to the Sorting Type selected the Output would vary.
- Number of Comparison would be obtained.
- Number of Swapping would be obtained.

- A table with the values tabulated would be obtained.
- Once the sample experiment is done, then **Refresh** button can be **clicked** to create New Samples.

Sorting Technique - Quick Sort

Programming Guidelines

This section guides the user to link his/her own code for Sorting Algorithm to NetSim.

Pre - Condition

User written program should read the value from the '**Input.txt**' in the temporary directory which is having input from the GUI at runtime

The output should be stored in '**Output.txt**' in the temporary directory for display.

User written program should return an integer value.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and, The results of the program should be written into the output file **Output.txt**.

Note: The naming of the input and the output file must be same as the text displayed in the Methodology screen

File Format

Input File Format	Output File Format
Ascending Order Sorting_Type=Quick Sorting_Order=Ascending	**type>index 1>index 2>** type 0 : specifies Positioning the data value index that ha type 1 : specifies the two index following it

Total_Number=3 Number_to_Sort=33,22,66 **Descending Order** Sorting_Type=Quick Sorting_Order=Descending Total_Number=3 Number_to_Sort=22,33,44	are being compared type 2 : specifies the two index following it are being swapped The following types are used in Quick Sort : type 3 : specifies the two index, where index 1 value is copied to index 2 value. type 4 : specifies the two index, where Position data index value of index1 is copied to the index 2 value. index 1 : index 1 is the index of array that points to the first data that is being swapped or compared. index 2 : index 2 is the index of array that points to the second data that is being swapped or compared. **Sample Output:** **Ascending:** 0>0>0> 1>0>2> 3>1>0> 4>0>1> 0>2>2> 0>0>0> **Descending:** 0>0>0> 3>2>0> 1>0>1> 4>0>2> 0>0>0> 1>0>1> 4>0>0> 0>1>1>

Interface Source Code

Interface source code written in C is given using this the user can write only the Sorting Algorithm inside the function fnQuicksort () using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ QuickSort.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista
- Properties --> find target in Windows XP.

Sample Scenarios:

Objective - To study the working of Sorting Techniques.

How to Proceed? -

The objective can be executed in NetSim using the programming exercise available, under programming user has to select Sorting Techniques.

Sample Inputs - In the Input panel the following steps need to be done,

- **SampleMode** should be **selected**.
- Select the Quick Sort as Sorting Type.
- Select the Sorting Order as either Ascending or Descending.
- Select the total number that has to be sorted. The values available are from **3** to **25**.
- Enter the Number Value in the field provided. The value entered should be within the range of **1** to **9999**.
- Then **Run** button need to be **clicked**. **Refresh** button can be used if new Inputs have to be given.

Output - The following steps are under gone internally,

- According to the Sorting Type selected the Output would vary.
- Number of Comparison would be obtained.
- Number of Swapping would be obtained.

- A table with the values tabulated would be obtained.
- Once the sample experiment is done, then **Refresh** button can be **clicked** to create New Samples.

Sorting Technique - Selection Sort

Programming Guidelines

This section guides the user to link his/her own code for Sorting Algorithm to NetSim.

Pre - Condition

User written program should read the value from the '**Input.txt**' in the temporary directory which is having input from the GUI at runtime

The output should be stored in '**Output.txt**' in the temporary directory for display.

User written program should return an integer value.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and the results of the program should be written into the output file **Output.txt**.

Note: The naming of the input and the output file must be same as the text displayed in the Methodology screen

File Format

Input File Format	Output File Format
Ascending Order Sorting_Type=Selection Sorting_Order=Ascending	**type>index 1>index 2>** type 0 : specifies Positioning the data value index that ha type 1 : specifies the two index following it are being compared

Total_Number=3 Number_to_Sort=88,77,66 **Descending Order** Sorting_Type=Selection Sorting_Order=Descending Total_Number=3 Number_to_Sort=55,11,22	type 2 : specifies the two index following it are being swapped The following types are used in Quick Sort : type 3 : specifies the two index, where index 1 value is copied to index 2 value. type 4 : specifies the two index, where Position data index value of index1 is copied to the index 2 value. index 1 : index 1 is the index of array that points to the first data that is being swapped or compared. index 2 : index 2 is the index of array that points to the second data that is being swapped or compared. **Sample Output:** **Ascending:** 0>0>0> 1>1>0> 0>1>1> 1>2>1> 0>2>2> 2>0>2> 0>1>1> 1>2>1> **Descending:** 0>0>0> 1>1>0> 1>2>0> 0>1>1> 1>2>1> 0>2>2> 2>1>2>

Interface Source Code

Interface source code written in C is given using this the user can write only the Sorting Algorithm inside the function select() using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ SelectionSort.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista
- Properties --> find target in Windows XP.

Sample Scenarios:

Objective - To study the working of Sorting Techniques.

How to Proceed? -

The objective can be executed in NetSim using the programming exercise available, under programming user has to select Sorting Techniques.

Sample Inputs - In the Input panel the following steps need to be done,

- **SampleMode** should be **selected**.
- Select the Selection Sort as Sorting Type.
- Select the Sorting Order as either Ascending or Descending.
- Select the total number that has to be sorted. The values available are from **3** to **25**.
- Enter the Number Value in the field provided. The value entered should be within the range of **1** to **9999**.
- Then **Run** button need to be **clicked**. **Refresh** button can be used if new Inputs have to be given.

Output - The following steps are under gone internally,

- According to the Sorting Type selected the Output would vary.
- Number of Comparison would be obtained.
- Number of Swapping would be obtained.
- A table with the values tabulated would be obtained.

- Once the sample experiment is done, then **Refresh** button can be **clicked** to create New Samples.

Spanning Tree – Borovska

Programming Guidelines

This section guides the user to link his/her own code for Spanning Tree using Borovska algorithm to NetSim.

Pre - Conditions

The user program should read the input scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%"** → **NetSim** → **"Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the Reading of the Inputs from the input file **Input.txt**.

Executing the required concept and

The results of the program should be written into the output file **Output.txt**.

Input File Format	Output File Format
Algorithm=Borovska No_of_Switches=3 No_of_Edges=3 Source_Switch=3,Destination_Switch=1, Distance=11, Source_Switch=1,Destination_Switch=2, Distance=22, Source_Switch=3,Destination_Switch=2, Distance=33,	'**Output.txt** ' file has the edges in the spanning tree, which is selected from the input edges. **Node1>Node2>Cost>** Example: 1>2>22> 1>3>11>

Interface Source Code

Interface source code written in C is given .Using this the user can write only the Borovska algorithm inside the function fnBorovska() using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ Boruvska.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista
- Properties --> find target in Windows XP.

Sample Scenarios:

Objective - To find a **SpanningTree** for a network using **Borovska's** algorithm.

How to Proceed? - The objective can be executed in **NetSim** using the programming exercise available. In the **Programming** Menu select**Spanning Tree**.

Sample Input

- **SampleMode** should be selected.
- **SelectBorovska** as **Algorithm** from the list available.
- **Select** the **Number** of **Switches**. **Select** 4 **Switches** as an **Input**.
- **Click** on 2 **Switches** to give the **distance** between them. Similarly connect all the **Switches** that are available in the network. The **Distances** between the **Routers** that needs to be given are as follows,
    - Switch 1 and Switch 2 → 67 Km
    - Switch 1 and Switch 4 → 46 Km
    - Switch 1 and Switch 3 → 89 Km
    - Switch 2 and Switch 3 → 78 Km
    - Switch 2 and Switch 4 → 99 Km
    - Switch 3 and Switch 4 → 56 Km

- **Click** on **Run** button to execute. **Refresh** button can be used if new **Inputs** have to be given.

Sample Output

- The **SpanningTreeTable** with the **Path** (**Source Switch** and **Destination Switch**) and **Distance** is obtained. Below is the Table that is obtained for the above inputs,

Spanning Tree Table

Path		Distance[KM]
1	2	67
3	4	56
1	4	46

- The "**Length** of the **SpanningTree** (**KM**)" would be given below in the output panel. The **SpanningTreePath** consists of **green** lines, whereas the **Non-SpanningTree** consists of **red** lines. Here in this Sample, "**Length** of the **Spanning Tree** (**KM**) → 169".

- Once the sample experiment is done, then **Refresh** button can be clicked to create new samples.

Spanning Tree – Kruskal

Programming Guidelines

This section guides the user to link his/her own code for Spanning Tree using Kruskal algorithm to NetSim.

Pre - Conditions

The user program should read the input scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%" → NetSim → "Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the Reading of the Inputs from the input file **Input.txt**.

Executing the required concept and

The results of the program should be written into the output file **Output.txt**.

Input File Format	Output File Format
Algorithm=Kruskal's No_of_Switches=3 No_of_Edges=3 Source_Switch=3,Destination_Switch=1,Distance=23, Source_Switch=1,Destination_Switch=2,Distance=34, Source_Switch=3,Destination_Switch=2,Distance=45,	'**Output.txt** ' file has the edges in the spanning tree, which is selected from the input edges. **Node1>Node2>Cost>** **Example:** 3>1>23> 1>2>34>

Interface Source Code

Interface source code written in C is given .Using this the user can write only the Kruskal algorithm inside the function fnKruskal() using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ Kruskal.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista
- Properties --> find target in Windows XP.

Sample Scenarios:

Objective - To find the **SpanningTree** for a network by using **Kruskal** algorithm.

How to Proceed? - The objective can be executed in **NetSim** using the programming exercise available. In the **Programming** Menu select**Spanning Tree**.

Sample Input

- **SampleMode** should be selected.
- **SelectKruskal** as **Algorithm** from the list available.
- **Select** the **Number** of **Switches**. **Select** 4 **Switches** as an **Input**.
- **Click** on 2 **Switches** to give the distance between them. Similarly connect all the **Switches** that are available in the network. The **Distances** between the **Routers** that needs to be given are as follows,
    - Switch 1 and Switch 2 → 67 Km
    - Switch 1 and Switch 4 → 46 Km
    - Switch 1 and Switch 3 → 89 Km
    - Switch 2 and Switch 3 → 78 Km
    - Switch 2 and Switch 4 → 99 Km
    - Switch 3 and Switch 4 → 56 Km
- **Click** on **Run** button to execute. **Refresh** button can be used if new **Inputs** have to be given.

Sample Output

- The **SpanningTreeTable** with the **Path** (**Source Switch** and **Destination Switch**) and **Distance** is obtained. Below is the **Table** that is obtained for the above inputs,

Spanning Tree Table

Path		Distance[KM]
1	4	46
4	3	56
1	2	67

- The "**Length** of the **SpanningTree** (**KM**)" would be given below in the output panel. The **SpanningTree Path** consists of **green** lines, whereas the **Non-SpanningTree** consists of **red** lines. Here in this Sample, "**Length** of the **SpanningTree** (**KM**) → 169"

- Once the sample experiment is done, then **Refresh** button can be clicked to create new samples.

Spanning Tree – Prims

Programming Guidelines

This section guides the user to link his/her own code for Spanning Tree using Prims algorithm to NetSim.

Pre - Conditions

The user program should read the input scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%" → NetSim → "Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the Reading of the Inputs from the input file **Input.txt**.

Executing the required concept and

The results of the program should be written into the output file **Output.txt**.

Input File Format	Output File Format
Algorithm=Prim's No_of_Switches=3 No_of_Edges=3	'**Output.txt**' file has the edges in the spanning tree, which is selected from the input edges.

Source_Switch=1 Source_Switch=3,Destination_Switch=1,Distance=44, Source_Switch=1,Destination_Switch=2,Distance=55, Source_Switch=2,Destination_Switch=3,Distance=66,	**Node1>Node2>Cost>** **Example:** 1>3>44> 1>2>55>

Interface Source Code

Interface source code written in C is given .Using this the user can write only the Prim's algorithm inside the function fnPrims() using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ Prims.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista
- Properties --> find target in Windows XP.

Sample Scenarios:

Objective - To find a **SpanningTree** for a network using **Prims** algorithm.

How to Proceed? - The objective can be executed in **NetSim** using the programming exercise available. In the **Programming** Menu select**Spanning Tree**.

Sample Input

- **SampleMode** should be selected.
- **SelectPrim's** as **Algorithm** from the list available.
- **Select** the **Number** of **Switches**. **Select** 4 **Switches** as an **Input**.
- **Click** on 2 **Switches** to give the **distance** between them. Similarly connect all the **Switches** that are available in the network. The **Distances** between the **Routers** that needs to be given are as follows,
    - Switch 1 and Switch 2 → 67 Km

- Switch 1 and Switch 4 → 46 Km
- Switch 1 and Switch 3 → 89 Km
- Switch 2 and Switch 3 → 78 Km
- Switch 2 and Switch 4 → 99 Km
- Switch 3 and Switch 4 → 56 Km

- **SourceSwitch** can be any 1 from list.Here, **SourceSwitch** is selected as"3".
- **Click** on **Run** button to execute. **Refresh** button can be used if new **Inputs** have to be given.

Sample Output

- The **SpanningTreeTable** with the **Path** (**SourceSwitch** and **Destination Switch**) and **Distance** is obtained. Below is the **Table** that is obtained for the above inputs,

SpanningTreeTable

Path		**Distance[KM]**
3	4	56
4	1	46
1	2	67

  - The "**Length** of the **SpanningTree** (**KM**)" would be given below in the output panel. The **SpanningTree Path** consists of **green** lines, whereas the **Non-SpanningTree** consists of **red** lines. Here in this Sample, "**Length** of the **Spanning Tree** (**KM**) → 169".
- Once the sample experiment is done, then **Refresh** button can be clicked to create New Samples.

Transmission Flow Control - Go Back N

Programming Guidelines

This section guides the user to link his/her own code for Go Back N to NetSim.

Pre - Conditions

The user program should read the inputted scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%" → NetSim → "Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and,

The results of the program should be written into the output file **Output.txt**.

Window Size:

The **Window Size** for Go Back N is **7**

Input File Format	Output File Format
Algorithm=Go_Back_N Data_File=C:\Users\P.Sathishkumar\ Documents\1 Th.txt> BER=0	Value1>Value2>Value3>Value4> **Types:** There are five types of formats of writing in the output file. Each format is written in specific condition, the types are explained below. The condition of writing the types is explained in the algorithm. **Type1:**

	Value1 - "CNT", Value 2 - output of the slidingcount function Value 3 - "FRAMES" Value 4 - "TRANSMIT" **Type2:** Value1 - "DT", Value 2 - Frame number, Value 3 - Frame's Source address, Value 4 - Frame's Destination address. **Type3**: Value 1 - "EV", Value 2 - Output of the intro_error function, Value 3 - Frame' Source address, Value 4 - Frame's Destination address. **Type4**: Value 1 - "ACK", Value 2 - "POS", Value 3 - Acknowledgement frame's Source Address, Value 4 - Acknowledgement frame's Destination Address. **Type5:** Value 1 - "DEL" Value 2 - count of frames being deleted Value 3 - "FRAME" Value 4 - "DELETED" **Note:** The above convention to write into the **'Output.Txt'** is **mandatory**. Values in Quotes**""** to be written into file **'Output.Txt'** as it is including Case. DT>Frame No>node1>node2> (DT denotes

	Data from node 1 to node 2) EV>Error Flag>node1>node2> (EV denotes Error Value - i.e If the above frame has error then set the error flag as 1 else set the flag as 0) ACK>POS>node2>node1> (Acknowledgement for that above frame received is sent to node 2 to node 1) Ex: CNT>1>FRAMES>TRANSMIT> DT>1>node1>node2> EV>0>node1>node2> ACK>POS>node2>node1> DEL>1>FRAME>DELETED>

Interface Source Code

Interface source code written in C is given using this the user can write only the Transmission Flow Control - Go Back N inside the function GoBackN () using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ GoBackN_TFC.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista
- Properties --> find target in Windows XP.

Sample Scenarios:

Objective - Send and receive the data using **Transmission Flow Control - Go BackN**.

How to Proceed? - The objective can be executed in **NetSim** using the programming exercise available. Under **ProgrammingMenu** select **Transmission Flow Control→Go Back N**.

Sample Input - By using the **InputPanel** that is available on the left hand side a **SampleScenario** is created. The Steps involved in creating are as follows,

- **SampleMode** should be selected.
- **SelectGoBackN** as **Algorithm**.
- **Create** a "**Datafile** (**.txt**)" and save it on the disk.
- **Select** the path of the above created "**Datafile** (**.txt**)" which could be **Maximum** of **100000bytes**.
- **SelectBit Error Rate** (**BER**) as "**10^-5**" from the drop down menu.

1. Then **Run** button need to be clicked.

Output - Output for the above **Sample** is as follows,

2. An **OutputTable** is obtained when **Run** button is clicked. In the **OutputTable**, **Transmittedframe** (**Data** and **Acknowledgement**) and their corresponding Counts is obtained. The **TotalCount** is also obtained. The table is given below,

Output

Total data frames to be transmitted - 24

Transmitted frame	Count
Data	30
Acknowledgement	5
	Total = 35

Note - The "Total data frames to be transmitted" and "Total count" in the **Output** table depends on size of the "**.txt**" file.

- Once the sample experiment is done, then **Refresh** button can be clicked to create new samples.

Transmission Flow Control - Selective Repeat

Programming Guidelines

This section guides the user to link his/her own code for Selective Repeat to NetSim.

Pre - Conditions

The user program should read the inputted scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%" → NetSim → "Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and,

The results of the program should be written into the output file **Output.txt**.

Window Size:

The **Window Size** for Selective Repeat is **7**

Input File Format	**Output File Format**
Algorithm=Selective_Repeat Data_File=C:\Users\P.Sathishkumar\ Documents\1 Th.txt> BER=0	Value1>Value2>Value3>Value4> **Types:** There are five types of formats of writing in the output file. Each format is written in specific condition, the types are explained below. The condition of writing the types is explained in the algorithm. **Type1:** Value1 - "CNT", Value 2 - output of the slidingcount function Value 3 - "FRAMES" Value 4 - "TRANSMIT"

	Type2: Value1 - "DT", Value 2 - Frame number, Value 3 - Frame's Source address, Value 4 - Frame's Destination address. **Type3**: Value 1 - "EV", Value 2 - Output of the intro_error function, Value 3 - Frame' Source address, Value 4 - Frame's Destination address. **Type4**: Value 1 - "ACK", Value 2 - "POS", Value 3 - Acknowledgement frame's Source Address, Value 4 - Acknowledgement frame's Destination Address. **Type5:** Value 1 - "DEL" Value 2 - count of frames being deleted Value 3 - "FRAME" Value 4 - "DELETED" **Note:** The above convention to write into the **'Output.Txt'** is **mandatory**. Values in Quotes**""** to be written into file **'Output.Txt'** as it is including Case. DT>Frame No>node1>node2> (DT denotes Data from node 1 to node 2) EV>Error Flag>node1>node2> (EV denotes Error Value - i.e If the above frame has error then set the error flag as 1 else set the flag as 0)

	ACK>POS>node2>node1> (Acknowledgement for that above frame received is sent to node 2 to node 1) Ex: CNT>1>FRAMES>TRANSMIT> DT>1>node1>node2> EV>0>node1>node2> ACK>POS>node2>node1> DEL>1>FRAME>DELETED>

Interface Source Code

Interface source code written in C is given using this the user can write only the Transmission Flow Control -Selective Repeat inside the function SelectiveRepeat () using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ SelectiveRepeat_TFC.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista
- Properties --> find target in Windows XP.

Sample Scenarios:

Objective - Send and receive the data using **Transmission Flow Control - Selective Repeat**.

How to Proceed? - The objective can be executed in **NetSim** using the programming exercise available. Under **Programming Menu** select **Transmission Flow Control→Selective Repeat**.

Sample Input - By using the **Input Panel** that is available on the left hand side a **Sample Scenario** is created. The Steps involved in creating are as follows,

- **Sample Mode** should be selected.
- **Select Selective Repeat** as **Algorithm**.
- **Create** a "**Data file** (**.txt**)" and save it on the disk.

- **Select** the path of the above created "**Data file** (**.txt**)" which could be **Maximum** of **100000 bytes**.
- **Select Bit Error Rate** (**BER**) as "**10^-5**" from the drop down menu.

1. Then **Run** button need to be clicked.

Output - Output for the above **Sample** is as follows,

2. An **Output Table** is obtained when **Run** button is clicked. In the **Output Table**, **Transmitted frame** (**Data** and **Acknowledgement**) and their corresponding **Counts** is obtained. The **Total Count** is also obtained. The table is given below,

Output

Total data frames to be transmitted -

Transmitted frame	Count
Data	
Acknowledgement	
	Total =

Note - The "Total data frames to be transmitted" and "Total count" in the **Output** table depends on size of the "**.txt**" file.

- Once the sample experiment is done, then **Refresh** button can be clicked to create new samples.

Transmission Flow Control - Stop and Wait

Programming Guidelines

This section guides the user to link his/her own code for Stop and Wait to NetSim.

Pre - Conditions

The user program should read the inputted scenario from text file named '**Input**' with extension txt.

The user program after executing the concept should write the required output to a file named '**Output'** with extension txt.

Note: The temporary directory is navigated through the following step.

Run →Type **"%temp%" → NetSim → "Input.txt"** and **"Output.txt"**

General Program Flow

The program begins with the *Reading of the Inputs* from the input file **Input.txt**.

Executing the required concept and,

The results of the program should be written into the output file **Output.txt**.

Input File Format	Output File Format
Algorithm=Stop_and_Wait Data_File=C:\Users\P.Sathishkumar\ Documents\1 Th.txt> BER=0	Value1>Value2>Value3>Value4> **Types:** There are five types of formats of writing in the output file. Each format is written in specific condition, the types are explained below. The condition of writing the types is explained in the algorithm. **Type1:** Value1 - "CNT", Value 2 - output of the slidingcount function Value 3 - "FRAMES" Value 4 - "TRANSMIT" **Type2:** Value1 - "DT", Value 2 - Frame number, Value 3 - Frame's Source address, Value 4 - Frame's Destination address. **Type3**: Value 1 - "EV", Value 2 - Output of the intro_error function, Value 3 - Frame' Source address, Value 4 - Frame's Destination address. **Type4**: Value 1 - "ACK",

	Value 2 - "POS", Value 3 - Acknowledgement frame's Source Address, Value 4 - Acknowledgement frame's Destination Address. **Type5:** Value 1 - "DEL" Value 2 - count of frames being deleted Value 3 - "FRAME" Value 4 - "DELETED" **Note:** The above convention to write into the **'Output.Txt'** is **mandatory**. Values in Quotes**""** to be written into file **'Output.Txt'** as it is including Case. DT>Frame No>node1>node2> (DT denotes Data from node 1 to node 2) EV>Error Flag>node1>node2> (EV denotes Error Value - i.e If the above frame has error then set the error flag as 1 else set the flag as 0) ACK>POS>node2>node1> (Acknowledgement for that above frame received is sent to node 2 to node 1) Ex: DT>1>node1>node2> EV>0>node1>node2> ACK>POS>node2>node1>

Interface Source Code

Interface source code written in C is given using this the user can write only the Transmission Flow Control - Stop and Wait inside the function stopandwait() using the variables already declared. To view the interface source code, go to

NetSim Installation path / src / Programming/ StopandWait.c

To find NetSim's Installation path right click NetSim icon and select

- Open file location in Windows 7
- Open file location in Windows Vista
- Properties --> find target in Windows XP.

Sample Scenarios:

Objective - Send and receive the data using **Transmission Flow Control**– **Stop and wait**.

How to Proceed? - The objective can be executed in **NetSim** using the programming exercise available. Under **Programming Menu** select **Transmission Flow Control →Stop and wait**.

Sample Input - In the Input panel the following steps need to be done,

- **SampleMode** should be **selected**.
- **Stop and Wait** needs to be **selected** for Algorithm.
- The path of the "**Data file (.txt)**" should be **entered**.
- **Bit Error Rate (BER)** should be **selected**.
- Then **Run** button need to be **clicked**. **Refresh** button can be used if new Inputs have to be given.

Output - The following steps are under gone internally,

- When the **Run** Button is **clicked** the Tool generates the Output in the following format,

Output

Total data frames to be transmitted

Transmitted frame	Count
Data	
Acknowledgement	

Total =

- Once the sample experiment is done, then **Refresh** button can be **clicked** to create New Samples.

User Accounts

This menu can be used to add new user, delete of existing users, and to set mode of working (Exam/Practice) of the user. It is typically used by professors / administrators to add student users, and then set the user to exam mode during assessments.

> ***Note**:As of v8 it is not yet possible to control set / modify user settings from a central location / server. All of the possible user accounts operations given below will have to be done in each client PC where NetSim is installed. Tetcos is exploring the possibility of providing centralized user accounts control in NetSim from v9 onwards.*

User Accounts can be reached through **Utilities →User Accounts**

The Administrator has to enter the correct password and click on OK button to enter into the User Accounts interface. This is used for verification purpose. The buttons available in User Accounts window are:

- Add User - This button is used for creating new users. The following fields have to be filled in,
    - Username - Name of the new user needs to be entered in the field provided.
    - Category - By default student is selected.
    - Mode - Either Practice / Exam mode needs to be selected.
    - Click on Accept / Cancel button to accept / cancel the changes.
- Delete User - This button is used for deleting the existing users.
- Change Mode - This button is used for changing the mode. There are two modes present in the software,
    - Practice Mode - Students will be able to get all the help that is associated with the software. Saved experiments can be reused. If this mode is selected, then the user would have access to
        1. Under Simulation all the saved experiments in Simulation when the user is in Practice mode can be “opened and reused”, “deleted if they are not required” and “Saved experiment can be used in the Analytics interface”.

2. Under Programming Sample all help documents such “Concepts, Algorithm, Pseudo Code and Flowchart” can be viewed.
3. Basics Menu,
4. NetSim Help.

- Exam Mode - Students will not be able to use the help associated with the software. Also saved experiments in the Practice mode cannot be reused in the Exam mode. If this mode is selected, then the user would have access to,
    1. Under Simulation all the saved experiments in Simulation when the user is in Exam mode can be “opened and reused”, “delete if they are not required” and “Saved experiment can be used in the Analytics interface”.
    2. Under Programming users cannotview “Concepts, Algorithm, Pseudo Code and Flowchart”.
    3. Basics Menu cannot be accessed,
    4. NetSim Help can be used.

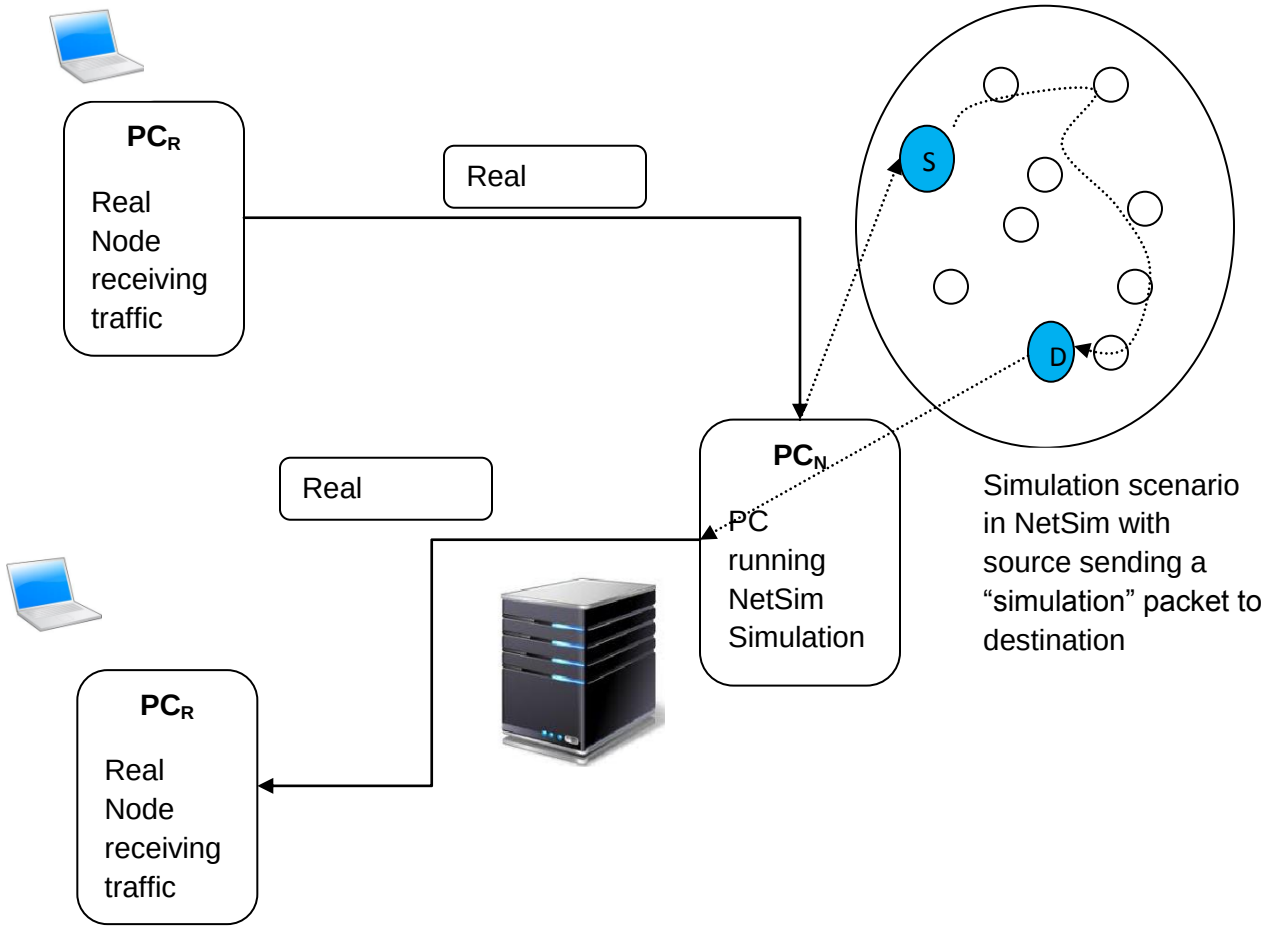
- Change Password - This button is used to change the password of the user which is selected currently in the left side of the User Accounts. Administrator can change the password of all the users, whereas other user (other than Administrator) can change his/her password only.
- Experiment Deletion Section: This section is used to delete the saved experiments in NetSim. Administrator can delete the saved experiments of all the users, whereas other user (other than Administrator) can delete his/her saved experiments only.

> ***Note:*** *Experiment Deletion Section can be used to delete the saved experiments in Legacy Networks, Advanced Wireless Networks– WiMAX and MPLS. Saved experiments in Internetworks, BGP Networks, Advanced Wireless Networks – MANET, Wireless Sensor Networks, Personal Area Networks, Cellular Networks, LTE Networks and Cognitive Radio Networks cannot be deleted through Experiment Deletion Section since these experiments are saved as configuration files in user specified locations.*

NetSim Emulator (Available only with Emulator Add on Module)

A network simulator mimics the behavior of networks but cannot connect to real networks. NetSim Emulator enabled users to connect NetSim simulator to real hardware and interact with live applications.

Emulation: How Simulation interacts with the real world



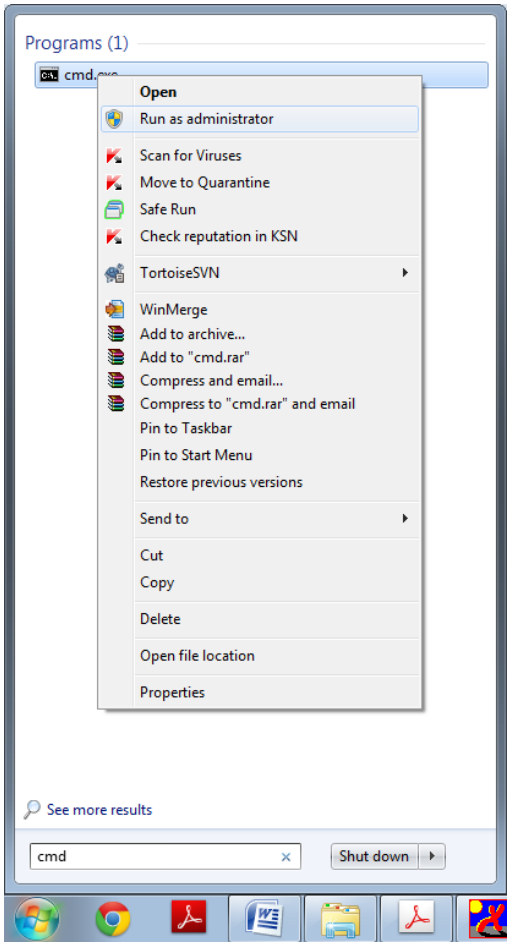
A real node, say PC_R sends data to the PC running NetSim, say PC_N. When ever a packet arrives at the interface of PC_N this packet is “modulated” into a simulation packet and sent from a source node (user selectable) in the network (user configurable) simulated to a destination node (again user selectable). Upon receipt of this packet at the destination, the packet is then “de-modulated” and sent back to a real destination node PC_R.

EmulationSet-up:

The ideal set-up to run emulation, would be to have three (3) PC's. One would be the real source, then another would run NetSim emulation server, and the third would be the real destination. Alternately, the set-up can also be managed with two PC's.One PC can be the source / destination and another PC would run the NetSim Emulation Server, and be the real destination / source.

1. **Setting IP configuration in Source and Destination (PC's NOT running the NetSim Emulation Server)**

   I. Open command prompt in administrative mode.

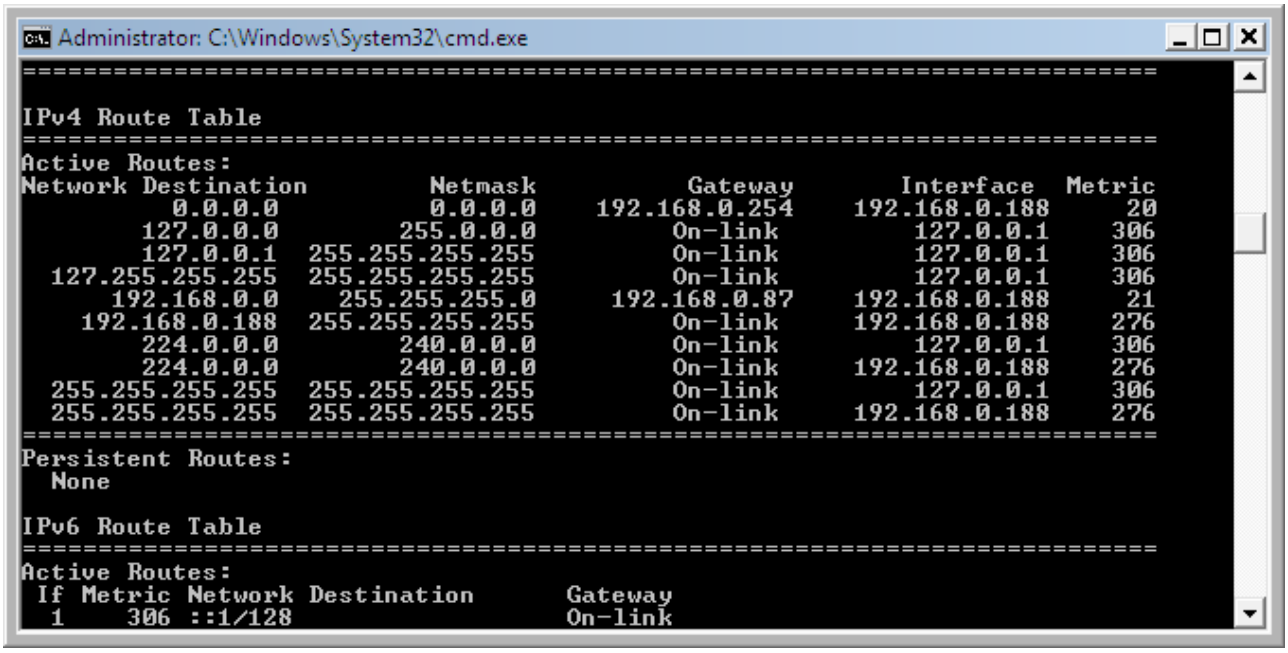


   II. Type command "route delete <Network Address>", then press Enter key. You will get "OK". For example if your IP address is 192.168.0.4 and the subnet mask is 255.255.255.0 then the network address is 192.168.0.0 (Got by performing a bitwise AND of the IP Address and the subnet mask)

III. Type command

"route add <Network Address>mask 255.255.255.0 <IP Address where NetSim Emulation serve is running> metric 1"(here the subnet mask is taken as 255.255.255.0). After execution, you will get "OK".

IV. Type command "netstat -r" to check if the IP configuration is done or not.



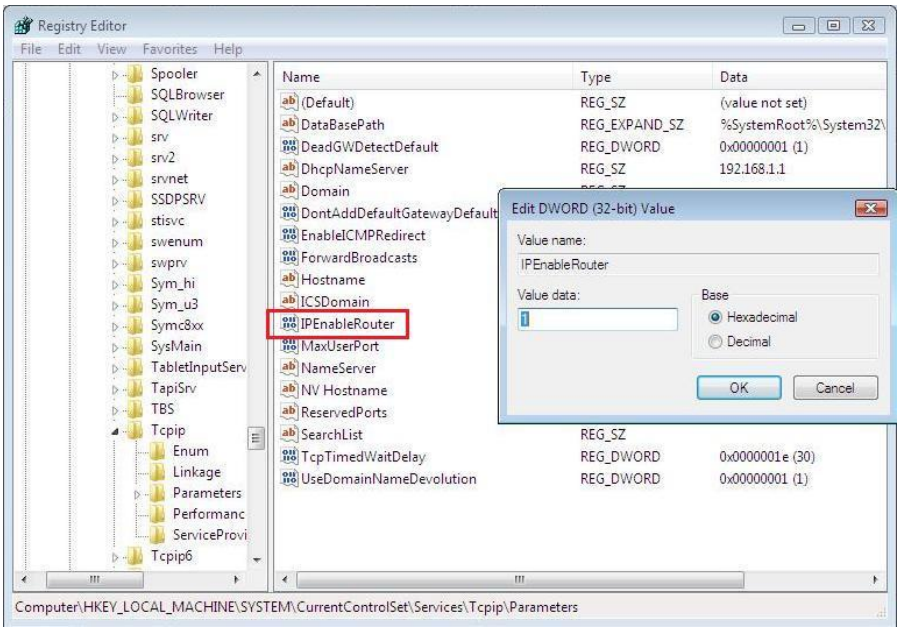
Note thatin the above screenshot, for the network 192.168.0.0, the gateway address assigned is 192.168.0.87(Address of the system where NetSim Emulation Server is running).

2. Setting in the PC running NetSim Emulation

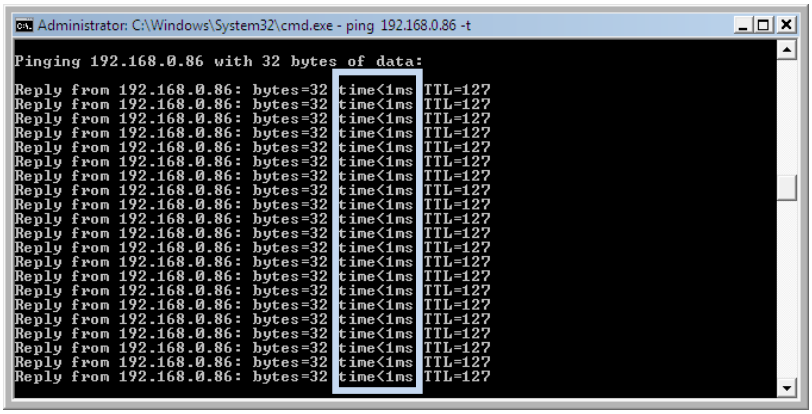
I. Open Registry Editor by going to Run -> regedit. Go to HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters

II. Set IPEnableRouter as 1

III. Restart System.



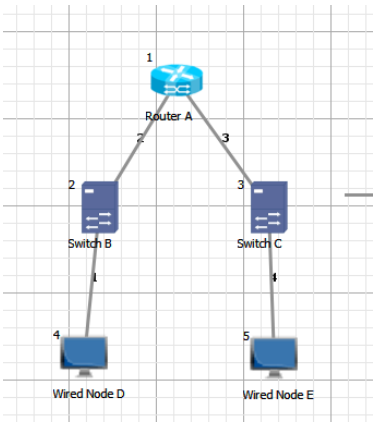
3. Ping the destination from the source and check the time duration.In this case you would notice the time to be<1ms.


Administrator: C:\Windows\System32\cmd.exe - ping 192.168.0.86 -t
Pinging 192.168.0.86 with 32 bytes of data:
Reply from 192.168.0.86: bytes=32 time<1ms TTL=127

Running Emulation in NetSim

1. **Example Application – ping**

I. Create a basicnetwork Scenario in any stack based protocol**(Internetworks, MANET, BGP, Cellular, WSN, Zigbee, CR, LTE**) in NetSim. Screenshot of a sample scenario in Internetworks is shown below



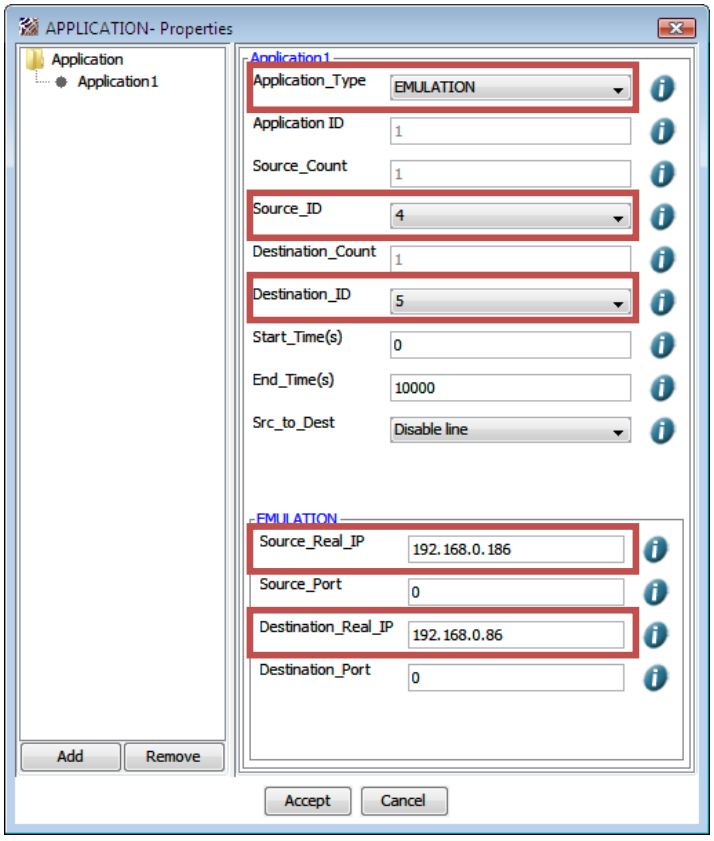
II. Click and drop the Application.

III. Right click Application->select Properties.

IV. In the Application Type select Emulation.

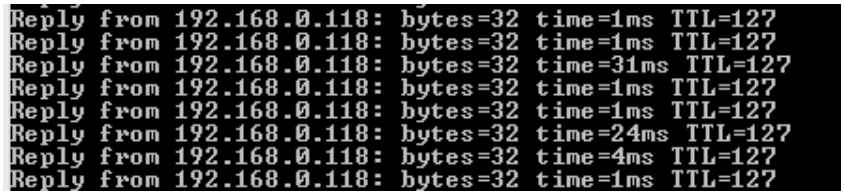
V. Select Source and Destination ID according to the network scenario.

VI. Change the Source and Destination IP address and click accept.



VII. Before running simulation, start pinging the Destination from Source using command "ping<Destination_IP> –t".

VIII. During Simulation you will notice that the present time duration is higher than the earlier ping results. This is because the network created in NetSim has link propagation delay of 5ms.


Reply from 192.168.0.118: bytes=32 time=1ms TTL=127
Reply from 192.168.0.118: bytes=32 time=1ms TTL=127
Reply from 192.168.0.118: bytes=32 time=31ms TTL=127
Reply from 192.168.0.118: bytes=32 time=1ms TTL=127
Reply from 192.168.0.118: bytes=32 time=1ms TTL=127
Reply from 192.168.0.118: bytes=32 time=24ms TTL=127
Reply from 192.168.0.118: bytes=32 time=4ms TTL=127
Reply from 192.168.0.118: bytes=32 time=1ms TTL=127

(NOTE: In case if no ping messages can be sent from source to destination, disable windows firewall and try again.)

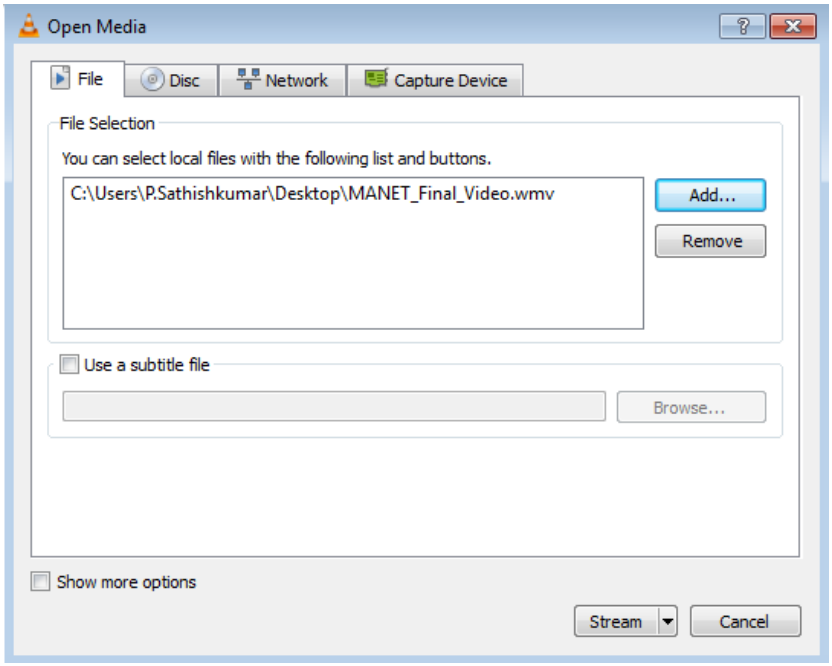
IX. The impact of the link propagation delay in NetSim Emulator is seen on a real packet

2. Example Application – Video

Steps at Source PC:

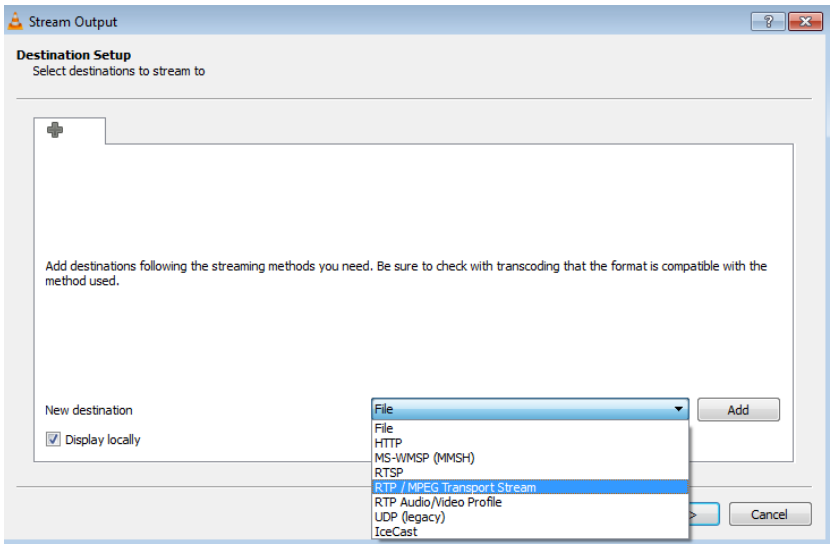
1. Open VLC Media player→Click Media menu→Select Stream Option.

2. Click add button then select the video which you want to play

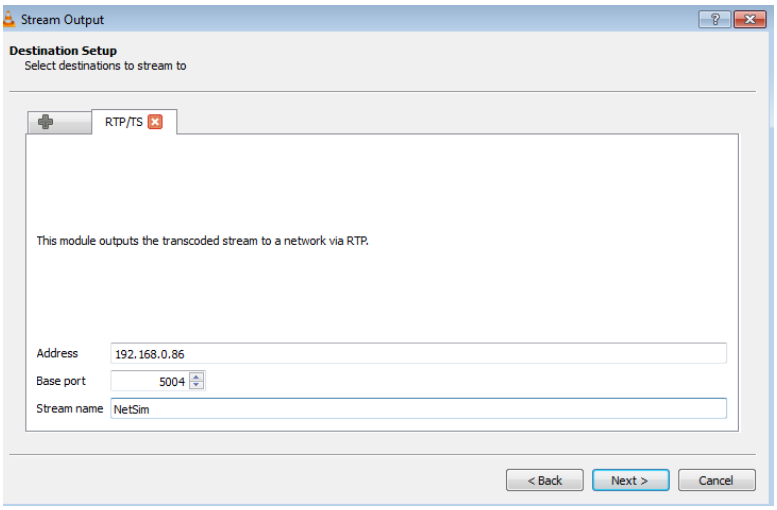


3. Click on Stream Option. Then click next button

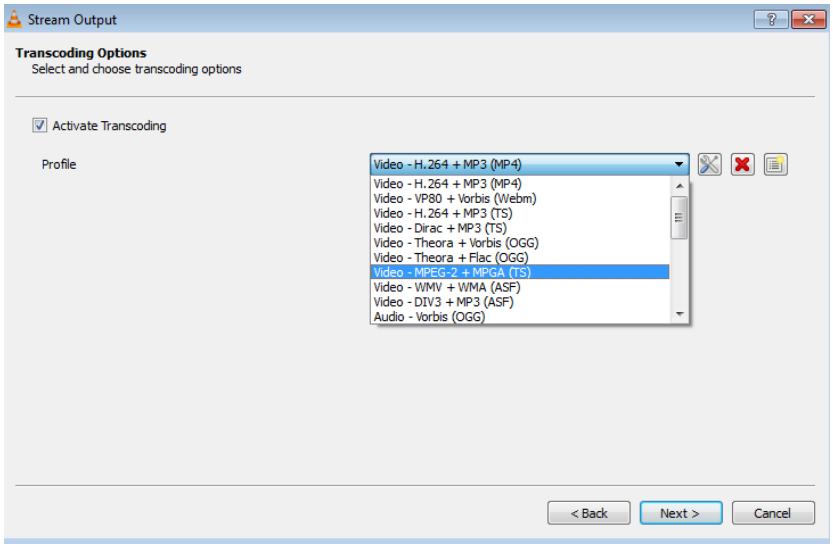
4. Enable the display locally checkbox. Then select the RTP / MPEG Transport Stream from the drop down list as shown in the below screen shot



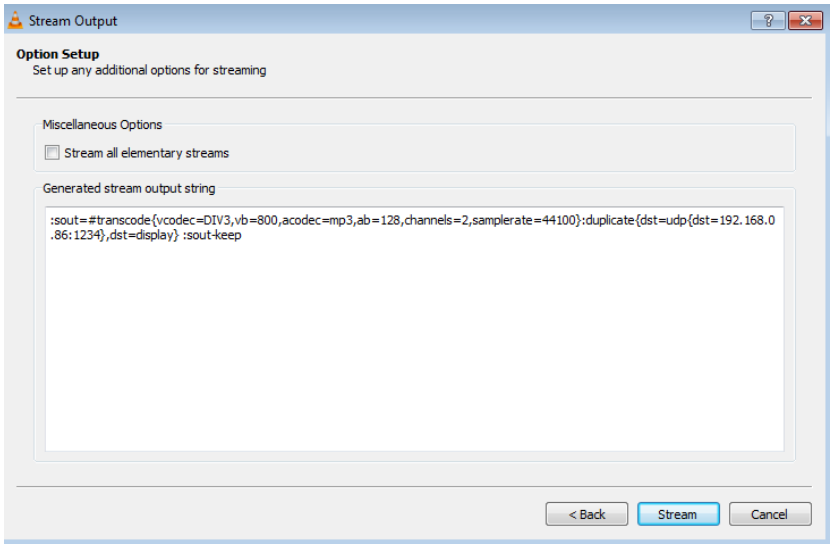
5. Click on Add Button. Then enter the Destination IP address in the Address field and enter a stream name (user defined) and click next button.



6. Select Video –MPEG-2 + MPGA (TS) option from the drop down list as shown in the below screen shot. Then click next button

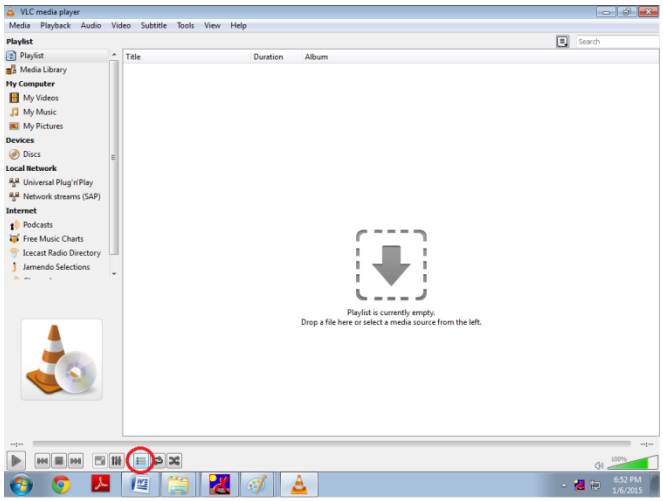

Stream Output
Transcoding Options
Select and choose transcoding options
Activate Transcoding
Profile
Video - H.264 + MP3 (MP4)
Video - H.264 + MP3 (MP4)
Video - VP80 + Vorbis (Webm)
Video - H.264 + MP3 (TS)
Video - Dirac + MP3 (TS)
Video - Theora + Vorbis (OGG)
Video - Theora + Flac (OGG)
Video - MPEG-2 + MPGA (TS)
Video - WMV + WMA (ASF)
Video - DIV3 + MP3 (ASF)
Audio - Vorbis (OGG)
< Back
Next >
Cancel

7. Click on Stream Button



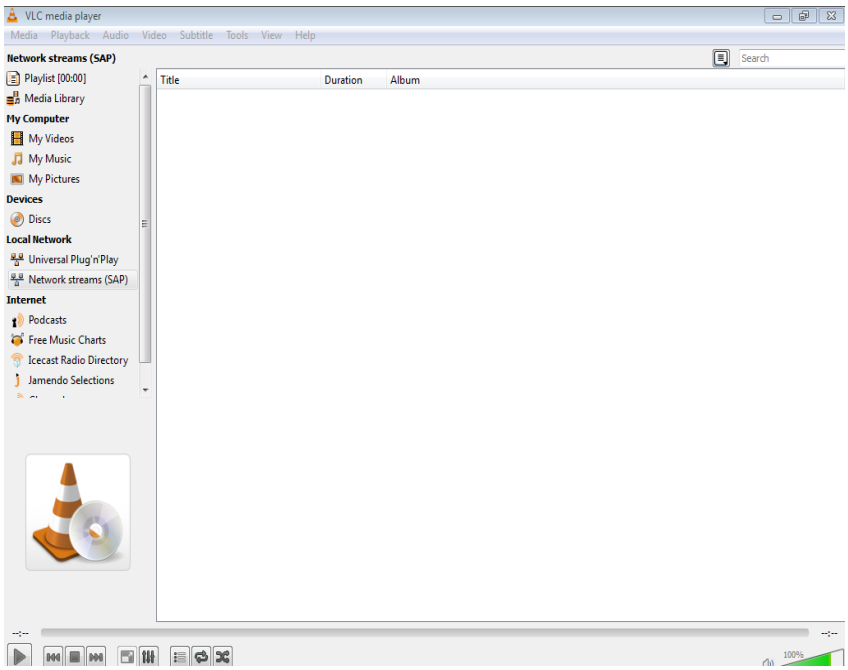
Steps at Destination PC:

1. Open VLC Media Player→ Click on Toggle Playlist icon as shown in the below screenshot.



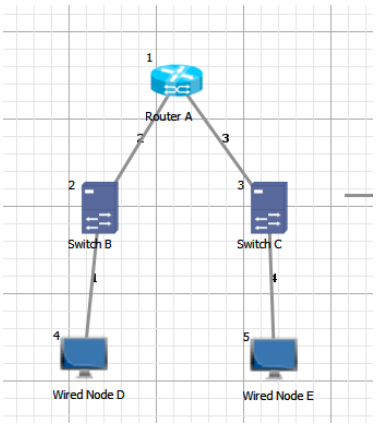
Toggle button is circled in red at the bottom of the screen shot

3. Double click on Network Stream(SAP) under local network. Then right click and play on the stream name that appears on the screen.



Steps at NetSim Emulation Server

I. Create a basic network Scenario in any stack based protocol **(Internetworks, MANET, BGP, Cellular, WSN, Zigbee, CR, LTE**) in NetSim. Screenshot of a sample scenario in Internetworks is shown below



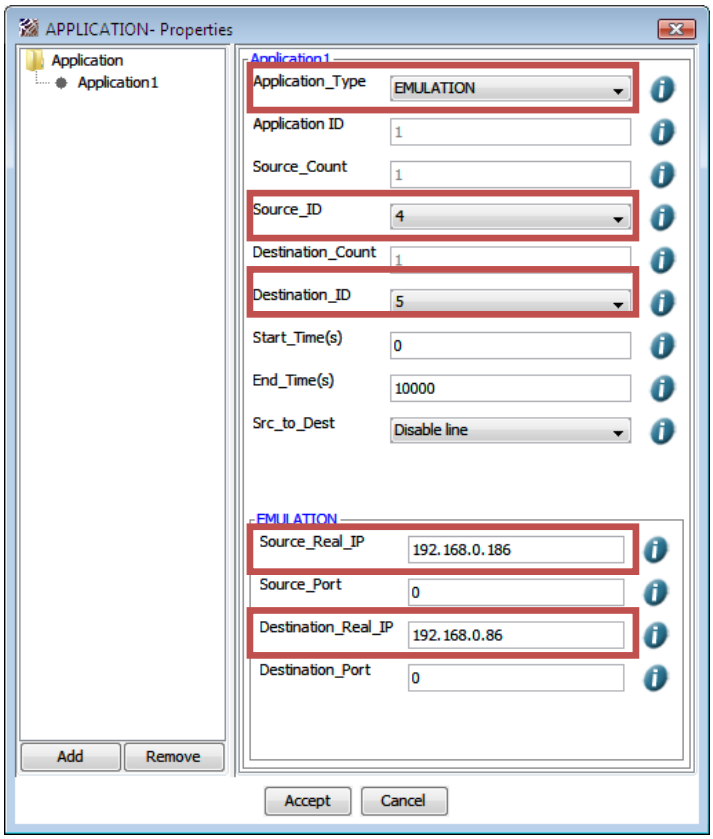
II. Click and drop the Application.

III. Right click Application->select Properties.

IV. In the Application Type select Emulation.

V. Select Source and Destination ID according to the network scenario.

VI. Change the Source and Destination IP address and click accept.



VII. Before running simulation, start streaming of the video from Source

VIII. Click Run Simulation

During Simulation you will notice a change in the quality of the video being played in the destination PC. This is because the network created in NetSim would have errors / delays etc. The impact of this loss / jitter / delay etc in NetSim Emulator is seen on a real video stream.